La Revue de l'Utilisateur PC

N° 76 mensuel Mars 1992-24 F

Le magazine professionnel en programmation PC

DU PROBLEME à la solution programmée...

WINDOWS 3.0
problème quotidien et perspectives

OPTIMISATION
des boucles en Assembleur

CLIPPER 5.01 :
la gestion de la mémoire

PRESENTATION
SFA.C et ARBRE_B.C

LE TRAITEMENT D'IMAGE
Mise au point d'un outil général de traitement et de manipulation d'images (1re partie)

CONVERSION
d'images graphiques : du format Bit-map au format PCX et vice versa

MARS 1992 - N° 76 - 24 F - Suisse 8 FS - Belgique 200 FB - Canada 6,25 $ Can.

M2537 - 76 - 24,00 F -RD

LOGICIELS DERNIERES VERSIONS
POUR PC, XT, AT, COMPATIBLES,
UNIX, XENIX ET MACINTOSH AUX

MEILLEURS PRIX

SOFTWORLD

17, Avenue Emile Zola, 75015 Paris. (Métro Javel)
Téléphone : (1) 40 59 02 99 Fax : (1) 45 79 95 55

Pour MACINTOSH

		Prix Public TTC	Notre prix TTC
BASES DE DONNEES			
4ème Dimension	f	~~7353~~	5885
File Force	f	~~3321~~	2250
FoxBase Mac Mono	æ	~~6820~~	5550
Omnis 5 *Blyth*	f	~~7110~~	6399
PAO / DESSIN			
Page Maker 4.0 *Aldus*	f	~~9607~~	7250
Canvas 3.0 *Deneba*	a	~~4140~~	3290

MAPLE V

Un outil intégral de calcul numérique et formel, de représentation graphique 2D ou 3D et de programmation symbolique procédurale de haut niveau pour Scientifiques, Mathématiciens. Version Mac ou autres systèmes.
Version Mac : 4270 TTC

TABLEURS			
Excel 3.0 *Microsoft*	f	~~4732~~	3295
WINGZ Avec graphe 3D	f	~~5325~~	4350
EDITEURS DE TEXTE			
Word 4 *Microsoft*	f	~~3546~~	2450
Formulator *(Mathématiques)*	a	~~1950~~	1490
INTEGRES			
Works *Microsoft*	f	~~2953~~	2250
LANGAGES			
MacNosy	a	~~ne~~	3950
Prolog 2.0 *Adv.A.I. Syst.*	a	~~3530~~	2950
Professional Programmer's Ext.	a	~~ne~~	2995
Prototyper (Générateur d'écran et de Codes sources)	a	~~ne~~	1995
QuickBasic *Microsoft*	a	~~1175~~	895
Smalltalk V/Mac *Digitalk*	a	~~1767~~	1503
Turbo Pascal *Borland*	f	~~1180~~	850
Think C 5.0	**a**	~~**2953**~~	**1695**
Think Pascal 4 .0	**a**	~~**2953**~~	**1695**
ZBasic 5.0 *Zedcor*	a	~~2022~~	1395
SCANNER ET O.C.R.			
Omnipage (O.C.R.)	a	~~11563~~	9750
Scanner couleur *Microtek*	a	~~29532~~	24900
UTILITAIRES			
Mac Tools Deluxe *Central P.*	a	~~1529~~	1095
SAM 3.0 (Antivirus)	a	~~1175~~	990
Virtual 030 pour Mac 68030	a	~~2360~~	1495

INGENIERIE

ACNAP Analyseur circuits AC	a	ne	4550
DCNAP Analyseur circuits DC	a	ne	2600
MICRO-CAP III (PC seulement)	a	ne	14950
SPP Traitement des signaux	a	ne	1950
Systat 5.0 Statistiques-Graphiques	a	ne	7950

MUSIQUE			
MacRecorder *Farallon*	a	ne	2150
Professional Composer *Mark Uni.*	a	~~6227~~	4950

a : Version en américain.
f : Version en français.
æ : en américain avec possibilité d'échange en français.
nc : non communiqué

TOUS NOS PRIX SONT TTC
PRIX INDICATIF AU 2 DECEMBRE 1991
Extrait de notre catalogue
Catalogue gratuit sur demande

POUR PC COMPATIBLES

		Public TTC	Notre TTC
ADA et Librairies			
Janus ADA Compiler +Tlkt	a	~~5871~~	4950
JanusADACompiler 386+Tkt	a	~~5871~~	4950
ASM et Librairies			
Btrieve 5.00C *Novell*	a	~~ne~~	5950
EMS Toolkit *Realtime H.*	a	~~ne~~	1350
Intel ASM-386 Macro ASM	a	~~ne~~	6000
Intel ASM-86 Soft.Dev.Kit	a	~~ne~~	7500
MASM 6.0 *Microsoft*	a	~~1530~~	1125
GX Effect + Source *Genus*	a	~~4732~~	3990
PCX Prog's Tlkit+Code Srce	a	~~ne~~	5995
GX Text + Source *Genus*	a	~~3546~~	2990
BASIC et Librairies			
Basic 7.1 *Microsoft*	a	~~4732~~	3795
Basic Develop. Tools *Sterli.*	a	~~ne~~	1175
Btrieve *5.0 Novell*	a	~~2906~~	2550
FABS PLUS *Computer C*	a	~~ne~~	2350
Graph Pak Pro *Crescent*	a	~~ne~~	1773
Logic Gem *Sterling Castle*	a	~~ne~~	1150
MicroHelp Toolbox	a	~~ne~~	1195
PDQ *Cresent*	a	~~ne~~	1295
Toolbox II *MicroHelp*	a	~~ne~~	1250
GX Effects + Source *Genus*	a	~~4732~~	3990
PCX Prog's Tlkit+Code Srce	a	~~ne~~	5995
GX Text + Source *Genus*	a	~~3546~~	2990
Realizer *Within Technology*	a	~~4685~~	3950
Sciences & Engin. Tools	a	~~ne~~	1195
PowerBasic *Spectra publish.*	f	~~1518~~	1139
Quick Basic 4.5 *Microsoft*	f	~~1175~~	895
QuickPak ProWindows *Cresc.*	a	~~2360~~	1990
QuickPak Pro QB 4.5 *Cresc.*	a	~~ne~~	2005
QuickPak Scientific *Crescent*	a	~~ne~~	1502
QuickScreen QB4.5 *Crescent*	a	~~ne~~	1502
QB Vol. 6 *MicroHelp*	a	~~ne~~	1295
Scientific SubroutineLibrary	a	~~ne~~	2250
Stay-res + *MicroHelp*	a	~~ne~~	1295
Visual Basic *Microsoft*	f	~~2360~~	1770
C, C++ et Librairies			
C Asynch Manager *Blaise*	a	~~ne~~	1890
C Compiler 6.0 *Microsoft*	a	~~4495~~	3250
C Tools Plus *Blaise Comput*	a	~~ne~~	1495
GFX Graphics Library	a	~~ne~~	2135
GFX Fonts & Menus Lib.	a	~~ne~~	2135
GFX Font Editor	a	~~ne~~	949
GFX Screen Dump	a	~~ne~~	949
GreenLeaf Data Windows	a	~~ne~~	4250
Halo *Pro Media Cybernetics*	a	~~ne~~	5396
High C 386 *MetaWare*	a	~~ne~~	8950
Lattice C *Lattice*	a	~~5812~~	4950
Meijin++ *Networks Integrat.*	a	~~3309~~	2790
Power Screen *Blaise Comput*	a	~~ne~~	1495
Power Screen *Blaise Comput*	a	~~ne~~	1246
Quick C 2.5 *Microsoft*	a	~~1175~~	895
Super Functions *Greenleaf*	a	~~ne~~	2950
Turbo C Tools *Baise Comp*	a	~~2123~~	1495
Turbo C++ 1.0 *Borland*	f	~~1773~~	1250
Turbo C++ Pro 1.0 *Borland*	f	~~3552~~	2350
Turbo C++ RuntimeLibrary	a	~~1773~~	1495
Win++ *Blaise Computing*	a	~~ne~~	2495
Zortech C++ 2.1 *Zortech*	a	~~1773~~	1450
Zortech C++ 2.1 Developper	a	~~5330~~	4530

		Public TTC	Notre TTC
COBOL			
Cobol 4.0 *Microsoft*	a	~~8290~~	5950
Cobol / 2 *Microfocus*	a	~~ne~~	18000
RM Cobol Dvt *Mac Farland*	a	~~ne~~	12500
DEBUGGEURS			
MultiScope - Windows	a	~~ne~~	3995
Periscope IV (25 MHz)	a	~~ne~~	25950
View232 *Blaise Computing*	a	~~ne~~	1890
EDITEURS			
Brief 3.0 *Solution Systems*	f	~~4495~~	2250
EC Editor *C Source*	a	~~ne~~	895
Epsilon *Lugaru Software*	a	~~ne~~	2250
KEdit *Mansfield Group*	a	~~ne~~	750
VEdit Plus *CompuView*	a	~~ne~~	1495
GENERATEURS			
Automated Programmer*KGK*	a	~~ne~~	4950
Matrix Layout pr Basic, C, Pascal	a	~~ne~~	1950
PCYACC Corporate*Abraxas*	a	~~ne~~	5500
INTELLIGENCE ART.			
PC SCHEME *Texas Instru.*	a	~~ne~~	1195
Smalltalk V *Digitalk*	a	~~ne~~	895
Smalltalk V/286 *Digitalk*	a	~~ne~~	1450
Smalltalk V/Windows *Digit.*	a	~~4732~~	4250
Communication Pak *Digitalk*	a	~~ne~~	425
EGA/VGA Pak *Digitalk*	a	~~ne~~	425
Goodies 1 ou 2 ou 3 *Digitalk*	a	~~ne~~	425
µuLisp 90 *Soft Warehouse*	a	~~ne~~	3558
PASCAL et LIBRAIRIES			
MTASK pro *Ramsi*	**f**	~~**2360**~~	**1950**
Pascal 4.0 *Microsoft*	a	~~4732~~	3695
Asynch Manager *Blaise C.*	a	~~ne~~	1750
Power Screen *Blaise C.*	a	~~ne~~	1490
Pascal View 232 *Blaise C.*	a	~~ne~~	1890
Quick Pascal *Microsoft*	**f**	~~**1175**~~	**895**
Turbo Pascal 6.0 *Borland*	**f**	~~**1773**~~	**1250**
Turbo Pascal Pro 6.0	**f**	~~**3552**~~	**2350**
AUTRES LANGAGES			
Actor *White Water Group*	a	~~4695~~	3785
Fortran 5.1 *Microsoft*	a	~~5990~~	3250
F77L-EM/32+386DOS Exten	a	~~16485~~	12928

PRO-CITE
Gestion bibliographique. Version US pour PC Compatible ou Macintosh
~~**4684,70 FF TTC**~~ **Notre prix: 3950 FF TTC**

DIVERS			
MapViewer *Golden Soft.*	a	~~3500~~	2975
Memory Commander	a	~~1120~~	950
MTask Pro 3.2 *Ramsi*	f	~~2965~~	2520
SOURCER+BIOS	**a**	~~**2075**~~	**1550**
Window 3.0 *Microsoft*	f	~~2360~~	1650
Windows Devel. Toolkit 3	a	~~4495~~	3295

UNIX
Systèmes / Langages / Fichiers / SGBD
Editeurs de Texte / Utilitaires
Nous contacter : (1) 40 59 02 99

		Public TTC	Notre TTC
BASES DE DONNEES			
Nantucket Clipper 5.0	f	~~11890~~	9250
dB XL *Wordtech systems*	f	~~4715~~	3850
Fox Base+ 2.10 *Fox Soft.*	f	~~9429~~	7500
Fox Pro Mono *Fox Soft.*	f	~~10615~~	8150
Genifer+Template CompilerFoxBas	a	~~ne~~	4195
Paradox 3.5 *Borland*	f	~~9963~~	7150
Super Base 4/Windows	f	~~7412~~	5411
CAO / DAO / PAO			
Design CAD 2D (3D +1000 Frs)	f	~~4685~~	3950
Page Maker 4.0 PC *Aldus*	f	~~9610~~	6750
Designer	f	~~11030~~	7530
COMPTA/GESTION			
Ciel! Compta-Gestion *Ciel!*	f	~~1156~~	925
Compta Saari Major *Saari*	f	~~16905~~	13950
INTEGRES			
Ability Plus Dual *Migent*	f	~~1927~~	1695
Works Windows *Microsoft*	f	~~2950~~	1995
TABLEURS			
Compile 12C Compiler WK.	a	~~ne~~	8950
Excel PC (286 et 386)	f	~~5918~~	3895
Lotus 123 Version 3.0	a	~~6750~~	4950
Multiplan 4 *Microsoft*	f	~~3309~~	2395
Quattro Pro 2.0 *Borland*	f	~~5924~~	4150
VP Planer + 3D *Paperback*	f	~~3440~~	2595
TRAITEMENT/ TEXTE			
Chiwriter (Scientifiques)	f	~~1235~~	1195
MathEdit (Mathématiques)	a	~~**2366**~~	**2366**
Sprint 1.5 *Borland*	f	~~2959~~	2050
Word 5.5 *Microsoft*		~~5325~~	3550
Word pour Window *Micros.*	f	~~5918~~	4150
Wordperfect 5.1 *Wordperf.*	f	~~5693~~	4595
UTILITAIRES / SYST.			
Analyser *Quaid*	a	~~2495~~	1950
CopyWrite *Quaid*	a	~~1175~~	695
Disk Explorer *Quaid*	a	~~1175~~	695
Dr DOS 6.0 *Digital Res.*	f	~~919~~	735
Norton Adv. Utilities 5.0	a	~~1779~~	1450
Norton AntiVirus	f	~~1175~~	950
PC Tools Deluxe 7.0	**a**	~~**2005**~~	**1395**
MATHS ET SCIENCES			
Derive 2.08 *Soft Warehouse*	**a**	~~**2965**~~	**1928**
EasyPlot V2.0	a	~~ne~~	3950
Grapher *(Graphiques scientif.)*	a	~~ne~~	2005
HiWire II *Wintek*	**a**	~~**11800**~~	**10031**
Maple V DOS-386/486	a	~~6274~~	4950
MathCAD Windows	a	~~4732~~	3786
Mathematica DOS-386/7 V2	a	~~9257~~	9257
Mathematica MS-Windows	f	~~10336~~	10336
Quartet *Softworld*	a	Tel	Tel
QuickPak Scientific *Crescent*	a	~~1767~~	1502
Surfer *(Surface, carte géo. 3D)*	a	~~6111~~	4950
Statistica *Statsoft*	a	~~9429~~	7675
SYSTAT 5.0 + SYGRAPH	a	~~ne~~	8995
TKSolver		~~ne~~	4685

386 DOS EXTENDER
286 DOS EXTENDER
Pour dépasser la limite 640 Ko du DOS
Prix: 4950 FF TTC pour Version 286 ou 386

BON DE COMMANDE
SOFTWORLD

17, Av. Emile Zola, 75015 Paris. (Métro Javel)
Téléphone : (1) 40 59 02 99 / Fax : (1) 45 79 95 55

Mon ordinateur est du type : (Macintosh) (PC Compatible)
Format de disquette : 3 1/2 ○ ou 5 1/4 ○
Je veux commander les produits notés ci-contre.
Mode de paiement :
○ Par Chèque ci-joint.
○ Par Contre remboursement +40 Frs (montant d'achat <5000 Frs)
○ Par Carte Bleue N°_____
Date d'expiration :_____
Date :
Signature : 1992

Nom :_____ Prénom :_____
Adresse :_____
Code postal :_____ Localité :_____
Téléphone :_____

Désignation	Quantité	Prix TTC
Frais d'envoi :		
Total TTC :		

Frais d'envoi recommandé obligatoire 40 Frs par 1000 Frs

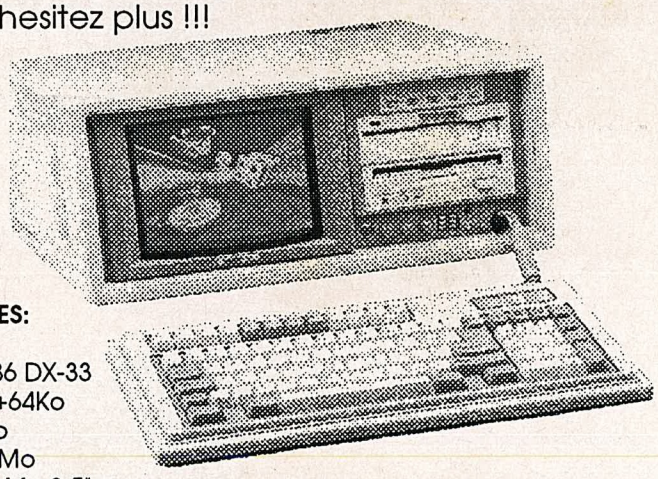
Aucun règlement ne sera encaissé avant l'expédition des marchandises

Nous fournissons la plupart des produits français et étrangers à des prix défiant toute concurrence et dans un délai record.
Plus de 3000 produits pour PC et MAC.
Applications verticales, Bases de données, D.A.O., Editeurs de Textes, Graphiques, Ingénierie, Intégrés, Langages, Mathématiques, Librairies, P.A.O, Statistiques, Tableurs, Utilitaires...

Heures d'ouverture
10h à 12h et de 14h à 19h
Fermé Samedi , Dimanche et Fériés

OFFREZ VOUS UN PORTABLE COULEUR !!

Enfin un vrai systeme portatif couleur qui vous suivra partout, doté d'une carte mère 80486, d'un écran CRT SONY, et de huit slots vous permettant de loger toutes vos cartes, n'hesitez plus !!!



CARACTERISTIQUES:

Processeur: 80486 DX-33
Cache: 8Ko+64Ko
Mémoire: 4 Mo
Disque dur: 210 Mo
Disquette: 1,44 Mo 3,5"
Clavier: 102 Touches ALPS
Ecran: SONY 9", 800x600, pitch 0,25
Sauvegarde Streamer (option): 250Mo

PRIX TTC : 22 500 Fr

VENEZ LE DECOUVRIR CHEZ FTC !!

CONFIGURATIONS MINI TOUR

Systemes complets avec 4Mo de mémoire, 1 lecteur 1,44Mo, 1 écran, 2 séries, 1//
Les configurations couleur comprennent 1 écran +1 carte 1024x768 512Ko

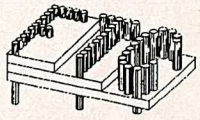
			Mono VGA	Super VGA
80386-25	Avec disque	40 Mo	7 948 F	9 844 F
sans cache		60 Mo	8 634 F	10 530 F
		80 Mo	9 066 F	10 962 F
		120 Mo	9 629 F	11 525 F
		210 Mo	11 386 F	13 283 F
80386-33	Avec disque	40 Mo	8 333 F	10 230 F
avec cache 64k		60 Mo	9 020 F	10 916 F
		80 Mo	9 451 F	11 348 F
		120 Mo	10 014 F	11 910 F
		210 Mo	11 772 F	13 668 F
80486-33	Avec disque	40 Mo	11 263 F	13 159 F
avec cache 64k		60 Mo	11 949 F	13 845 F
		80 Mo	12 381 F	14 277 F
		120 Mo	12 943 F	14 840 F
		210 Mo	14 701 F	16 597 F

PRIX TTC

Extensions Mémoires pour AST, Compaq, Dell, Epson, IBM, HP, Commodore, NCR, Panasonic, Toshiba, Zenith, Sharp, TI, Tandon **aux prix les plus bas...**

!! CHANGEZ VOTRE CARTE MERE !!
Transformation de votre systeme dans notre atelier... Consultez nous.

Lecteurs code barre	Nous consulter
Bibliothèque CD-ROM	Nous consulter



FTC
71 rue Vasco de Gama
75015 PARIS
Tél: 45 33 52 30 / 52 51
Fax: 45 33 50 55
Lundi à Vendredi
9H30 - 13H
14H00-18h00

PRIX TTC

ECRANS COUL.:	
VGA 14" HYUNDAI	2 750 F
VGA 14" EIZO 9065S	5 770 F
VGA 16" EIZO 9080i	9 990 F
VGA 17" Trinitron EIZO	15 920 F
VGA 20" EIZO 9400i	21 260 F
ECRANS MONO.:	
VGA 14"	990 F
VGA 17" EIZO	7 470 F
VGA 21" EIZO	11 475 F
IMPRIMANTES LASER BROTHER	
(Garantie sur site 2ans)	
HL-4V (compatible HPIII)	9 120 F
HL-4PS (PostScript)	13 000 F
HL-8D	24 900 F
HL-8V	13 000 F
HL-8PS	23 900 F
Cartes mère:	
286-16	750 F
386 SX-25	1 450 F
386 DX-25	2 550 F
386 DX-33 cache 64K	2 990 F
486 DX-33 cache 64K	5 950 F
Cartes d'affichage:	
VGA 1024x768 Trident 1Mo	850 F
VGA 1024x768 Trident 512K	670 F
VGA 1024x768 Trident 256K	580 F
Divers	
Claviers 102 touches	285 F
Boîtier Mini Tour	610 F
Coprocesseur 287 - 6-20 Mhz	850 F
Coprocesseur 387 - 25 Mhz	1 100 F
Coprocesseur 387-33 Mhz	1 650 F
Souris	220 F
Joystick	240 F
Sound Blaster Pro	1 990 F
Kit CD-ROM+Sblaster pro	3 990 F
Kit Multimédia 5 CD	1 390 F
10 Disquettes 1,2 Mo	55 F
10 Disquettes 1,44 Mo formattées	72 F
Mémoires de masse:	
FDD 1,2 Mo	540 F
FDD 1,44 Mo	490 F
HDD AT-bus 40 Mo	1 650 F
HDD At-bus 80 Mo	2 600 F
HDD AT-bus 120 Mo	3 150 F
HDD AT-bus 210 Mo	5 100 F
Streamer Colorado 250 Mo	3 700 F
Bande Streamer	320 F
Scanners à main	
A4GRAY 400dpi/256 niveaux de gris *(105mm, +Logiciels)*	1 400 F
A4COLOR 100dpi/4096 couleurs *(64mm, +Logiciels)*	2 450 F
A4COLOR 100dpi/262 144 couleurs *(105mm, +Logiciels)*	2 960 F
Carte FAX	
Intel SatisFaxion (9600/2400 Bps - µp 64K cache FaxIt, Logiciel WINDOWS 3)	3 900 F

SOMMAIRE

4

WINDOWS 3.0 PROBLEME QUOTIDIEN ET PERSPECTIVES

Dans l'absolu, WINDOWS 3.0 lancé en mode 386 étendu sait faire tourner sur toutes les machines les applications développées pour lui de même que les applications purement DOS.
Par Daniel VASSEUR

14

LE TRAITEMENT D'IMAGE MISE AU POINT D'UN OUTIL GENERAL DE TRAITEMENT ET DE MANIPULATION D'IMAGES (1re PARTIE)

Modifier la luminance d'une image, la réduire d'un facteur non entier, y appliquer des filtres de couleurs .
Par Jean-Michel SPINER

32

CLIPPER 5.01 : LA GESTION DE LA MEMOIRE

Les techniques de gestion de la mémoire pendant l'exécution ont beaucoup changé depuis Clipper 87. Le VMM (Gestionnaire de Mémoire Virtuelle) a fait son apparition. De nombreuses options de compilation et d'édition de liens ont bien entendu suivi. Voici un tour d'horizon, non exhaustif, de ces nouvelles fonctionnalités.
Par Dominique BRUYERE

38

CONVERSION D'IMAGES GRAPHIQUES : DU FORMAT BIT-MAP AU FORMAT PCX ET VICE VERSA

Travailler en mode graphique est devenu réellement une ''mode'' suivie par tout le monde à un tel point que la plupart des logiciels et des applications se convertissent ou s'intègrent sous l'environnement WINDOWS
Par Salah RIMANI

46

OPTIMISATION DES BOUCLES EN ASSEMBLEUR

En dépit du développement croissant des langages de haut niveau, de la vogue des L4G (Langage de 4e Génération),
Par Paul FRANCESCHI

50

DU PROBLEME A LA SOLUTION PROGRAMMEE...

Le poete Nicolas BOILEAU a écrit : ''Avant donc d'écrire apprenez à penser''. Cette citation devrait servir de règle d'or aux programmeurs, car combien sont-ils à réfléchir pleinement à leur application avant de commencer à s'abîmer les yeux devant leur écran.
Par Jean-Christophe RIAT

56

RECTIFICATIF

Une erreur s'est glissée dans le listing CALC.C figurant dans La Revue de l'Utilisateur PC n° 74, (Novembre-Décembre 1991) ne tourne pas.
Voici le nouveau programme.
Par Jean-Yves DEFRAYE

La Revue de l'Utilisateur PC

Le magazine professionnel en programmation PC

Mensuel 24 F
2, rue d'Amsterdam
75009 Paris
Tél. : (1) 42.81.54.27.
Service abonnement :
Tél. : (1) 42.81.54.27.

Directrice de la Publication :
Liliane ERRERA

Editeur :
Roger CHRISTOPHE

Rédacteur en chef :
Daniel VASSEUR

Directeur de fabrication :
Michel MILLET

Directrice de publicité :
Sardie ERRERA
Tél. : (1) 42.67.93.80.

Edité par SOFTin Communications (France) Sarl
R.C. Paris B 329 834 592
N° Siret
329 834 592 000 19
Commission Paritaire (en cours)
Dépôt légal :
1er trimestre 1992

CCP Paris 8850 84 D

Impression :
SNIL
93600 Aulnay-sous-Bois

Composition :
SOFTin Communications

SUMMER WINDOWS Frs 4 900 HT
Un environnement de programmation graphique
Gestion d'Interface Graphique
Dépassement des limites de taille du DOS

DGT Frs 3 250 HT
pour incorporer images,
photos dans vos champs mémos

~~10.500 FHT~~ H.T.
5900 Frs
NANTUCKET 5.01
Version française
+ Port 100 Frs

ProVision WINDOW Un moyen simple de créer des applications CLIPPER norme CUA. Initiez vous aux objets et à la programmation évènementielle. ProVision se charge de gérer l'interface utilisateur (menus, souris, windowing : fenêtre de taille modifiables, icônes.) **Frs 2 250 HT**

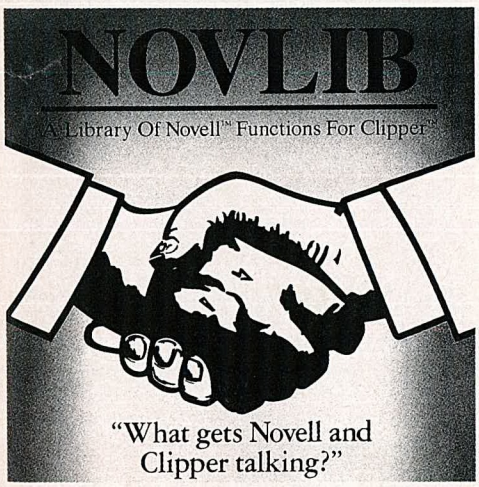
BARCODE LIBRARY Frs 3 750 HT.
CLTEXT Un vrai traitement de texte pour une application sous CLIPPER Frs 2 250 HT.
CLREPORT Un éditeur de rapport simple et puissant Frs 2 100 HT.
BLINKER V.F. Gain de mémoire, 15 fois plus rapide LE LINKER indispensable. Frs 2 850 HT.
NOVLIB V.F. Les 190 fonctions pour NOVELL Frs 2 890 HT.
TELEPATHY V.F. La télécommunication et CLIPPER Frs 2650 HT.
AQUARIUM Journal électronique sur CLIPPER 5.01 entre 60 et 80 pages mensuelles.
CLIPWKS de Lotus ou QUATTRO VERS CLIPPER ET VICE VERSA avec moins de 10 k Frs 1 750 HT.

Port et emballage 100 F sur nos produits.

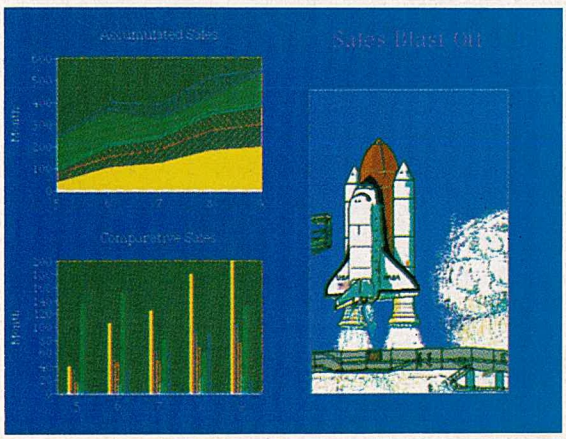
SEMINAIRE DANS VOTRE REGION : 7 personnes maximum

- CLIPPER 5.01 DEBUTANT ET EXPERT
- CLIPPER et la philosophie Windows
- CLIPPER ET LA TELECOMMUNICATION
- CLIPPER PAR L'OBJET
- RESEAU NOVELL
- C++ TURBO PASCAL

FORMATION SUR SITE
Demandez notre catalogue

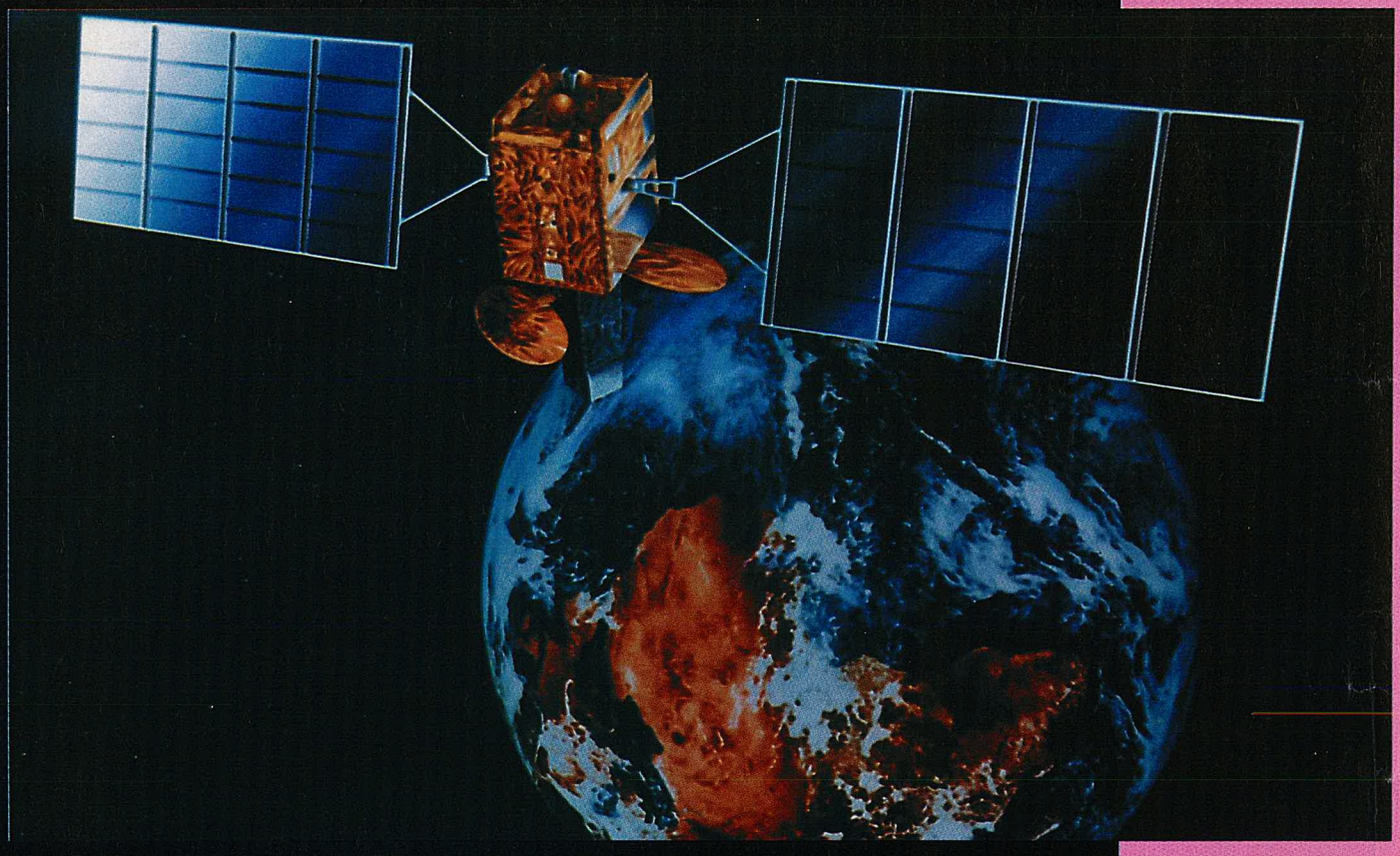


NOVLIB V.F. Frs 2 890 HT.
Les 190 fonctions pour NOVELL



FLIPPER Frs 2 650 HT
La lib graphique la plus complète
pour votre application CLIPPER

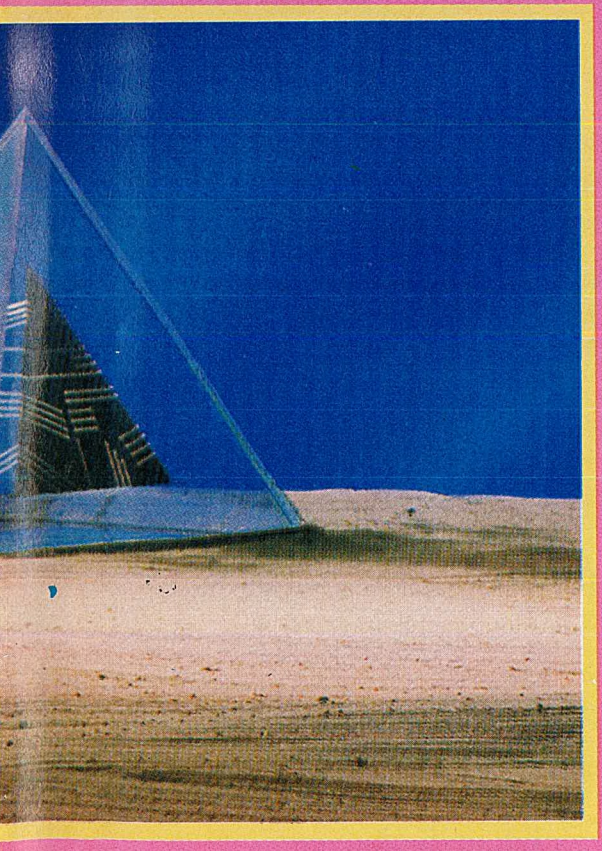
L.D.Q.S
5 rue Eugene Emanuel 06000 NICE. FRANCE.
☎ : (33). 93.16.09.12 - fax : (33). 93.88.08.90.



Dans l'absolu, WINDOWS 3.0 lancé en mode 386 étendu sait faire tourner sur toutes les machines les applications développées pour lui de même que les applications purement DOS.

Daniel VASSEUR

WINDOWS 3.0 problème quotidien et perspectives



Pourtant, il faut parfois réaliser des prouesses de paramètrage de l'environnement pour permettre à certaines applications de fonctionner correctement dans les fameuses boîtes DOS.

Partons d'un cas pratique fréquemment rencontré et examinons ensemble la solution appliquée aux applications générées à l'aide du compilateur CLIPPER 87.

Posons le décor et imaginons que nous travaillons sur une machine de type 386 équipée d'un simple pilote XMS qui gère les 4 MO mémoire étendue.

Nous disposons d'une application développée en NANTUCKET 87 et qui fonctionne parfaitement bien sous DOS. Avec le temps, cette application a grossi et prend de plus en plus de place en mémoire.

Tant qu'elle s'exécute sous DOS pur, aucun problème, sauf peut-être qu'il faut désactiver les éventuels programmes résidants trop gourmands de mémoire conventionnelle. Etant donné qu'il n'y a aucune disponibilité EMS (Expanded Memory Specification), l'application ne dispose que de mémoire conventionnelle, puisque les exécutables CLIPPER intègrent par défaut une gestion des buffers de réindexation en mémoire paginée dès lors qu'un gestionnaire de ce type est détecté.

Le paramètrage général de l'application se fait à travers la variable CLIPPER introduite dans le string d'environnement grâce à l'ordre :

SET CLIPPER = F40

qui a pour but d'indiquer l'utilisation de 50 fichiers (permis depuis DOS 3.30).

En réalité, pour simplifier, il existe un fichier batch qui automatise l'ensemble des opérations et qui présente l'aspect suivant :

```
C:
CD \APPLI
SET CLIPPER=F40
APPLI.EXE
CD \
```

Pour gérer la totalité du string d'environnement sans rencontrer de difficulté, l'ordre SHELL a été inséré dans le fichier CONFIG.SYS de la façon suivante :

SHELL = C:\COMMAND.COM /E:512 /P

de sorte que l'espace environnement puisse supporter 512 octets de données.

Sur cette même machine, WINDOWS 3.0 a été installé et une icône de chargement de l'application NANTUCKET a été définie : elle se contente de

recouvrir le lancement du fichier batch que je viens de décrire.

Et c'est là que les problèmes surviennent : l'application semble fonctionner de façon erratique, des plantages aléatoires se produisent sans que l'on puisse détecter leur origine.

Pour comprendre la raison de ces plantages, il nous faut faire un petit tour à l'intérieur de WINDOWS et bien comprendre les mécanismes de création d'une boîte DOS.

Pour permettre le lancement d'un programme DOS, WINDOWS en mode 386 étendu crée une machine virtuelle 8086 partiellement occupée par l'ensemble des programmes chargés avant le lancement de l'environnement graphique, plus une petite partie de WINDOWS de taille variable selon le paramètrage de la machine. J'ai déjà rencontré cette partie résiduelle de WINDOWS pour des tailles aussi différentes que 2 400 ou 90 000 octets.

Nous reviendrons une autre fois sur le paramètrage système de la machine pour un rendement optimal.

Dans la boîte DOS créée, WINDOWS recharge une copie du processeur de commande du DOS, le célèbre COMMAND.COM que l'on ne présente plus, et qui se rechargera pour sa plus petite partie soit environ 4 KO en version 3.30 et un peu moins en version 5.0.

Tout pourrait sembler correct, et pourtant ce paramètragre simpliste est source de difficultés.

En rechargeant une simple couche COMMAND.COM sans autre indication, WINDOWS ne peut recréer un environnement identique au précédent pour ce qui concerne l'espace mémoire disponible pour les variables d'environnement.

En effet, dans ce cas, la nouvelle couche COMMAND.COM calcule la taille de l'espace environnement initial en nombre d'octets réellement utilisés et arrondit le résultat au nombre de paragraphes immédiatement supérieur pour trouver la taille définitive.

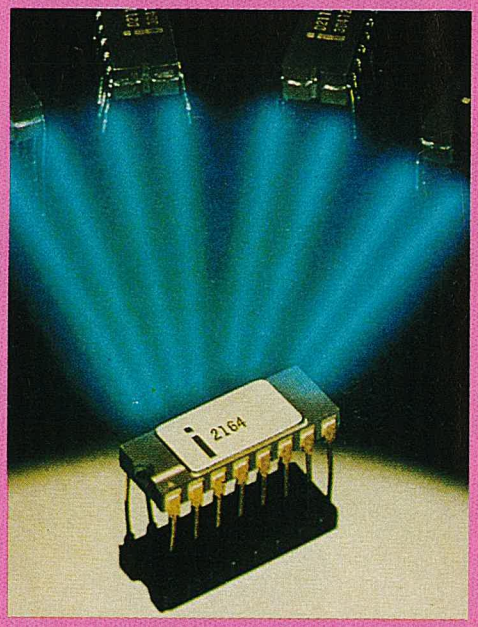
Il devient alors très difficile d'introduire toute nouvelle variable dans l'espace environnement, et le paramètrage du nombre des fichiers utilisés par l'application NANTUCKET a de fortes chances de ne pas aboutir. Conséquence immédiate, des erreurs du run-time pour "Trop de fichiers ouverts" ne tardent pas à pointer le bout du nez.

Une première solution — qui fonctionne — consiste à placer toutes les variables requises dans le string d'environnement du DOS avant le chargement de WINDOWS.

Efficace mais inutilisable dès lors que différentes applications au paramètrage différent sont susceptibles d'être utilisées sur la même machine.

La solution existe pourtant, et elle n'est pas limitée aux seules applications NANTUCKET : elle s'adresse à tous les programmes qui utilisent des variables placées dans l'espace environnement du DOS, méthode de tous temps préconisée et largement appliquée par MICROSOFT.

La solution donc, passe par la rédaction d'un fichier PIF tel que reproduit en Figure n° 1.



A l'aide de l'utilitaire PIFEDIT, il suffit de créer un fichier de paramètrage qui contient COMMAND.COM comme nom de l'application à lancer et de lui passer en paramètres optionnels le désormais classique /E:512 qui définit l'espace environnement à 512 octets ainsi que l'ordre de chargement (para-

NOUVEAU !

AG - OUTIL

Outil de programmation
pour le développement en CLIPPER ©

• Réalisation instantanée:
64 types de menus possibles,
Pages écrans,
Mise à jour rapide de données
sous forme de liste,
Formats d'impression

• Macro environnement:
Réduit de 50% le temps de programmation

• Entièrement paramétrable:
Souplesse et sur mesure

• Protection d'accès (10 niveaux)

• Multifichiers

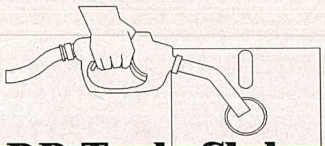
Pour tout renseignement ou documentation
appeler le 43.96.93.18 ou fax. 43.96.07.44

CB informatique
30, avenue du Maréchal
de Lattre de Tassigny
94410 SAINT-MAURICE
Tél. 43.96.93.18 - Fax : 43.96.07.44

CB informatique
NORD PAS DE CALAIS
Tél. 21.01.09.12

Maintenant essayez, gratuitement, quelques uns des meilleurs logiciels du monde avant de les acheter

- ☞ vous avez à disposition l'une des plus grandes bibliothèques de programmes en libre essai pour Win 3.0, mais aussi des logiciels techniques, éducatifs, des jeux, et une vaste librairie de programmation pour dBase, Clipper 5.01,QB 4.5, C/C++, TPascal 6.0 & Visual Basic
- ☞ vous ne payez que nos frais de diffusion : **25F ttc en 5"1/4 et 27F ttc en 3"1/2**
- ☞ vous recevez des disquettes gratuites 1 pour 7, 2 pour 10, 5 pour 20 et 10 pour 30 !
- ☞ vous ne prenez aucun risque : **si vous n'êtes pas satisfait, vous serez remboursé**
- ☞ vous avez les dernières versions, nous téléchargeons des USA les mises à jour chaque nuit
- ☞ en passant commande vous recevez gratuitement notre catalogue et notre lettre mensuelle qui vous tient au courant des derniers logiciels shareware sortis en France et aux USA
- ☞ vous recevez des disquettes auto-décompactables contrôlées par 2 anti-virus récents
- ☞ votre commande vous sera expédiée dans les 24 heures, 6 jours sur 7, toute l'année.



DP Tool Club

agrée par l'Association of Shareware Professionals
102 rue des fusillés 59657 Villeneuve d'Ascq
tél: 20.56.55.33 (de 14h à 17h) Fax: 20.56.55.25

Nouveautés

❑ **PC Index 4.0** 1879
Génère des index à partir de fichiers textes, avec gestion des inclusions et exclusions.

❑ **Secret Agent & Bolo Ball** 875
Le dernier né des jeux d'arcade de chez Apogee et un bon jeu de stratégie de W Soleau. Les habitués de DP Tool Club apprécierions. Pour écran EGA/VGA.

❑ ❑ **Galactix** 876AB
Un superbe jeu d'arcade, avec sortie son possible sur AdLib, SB ou sur HP interne. Pour AT, disque dur et VGA.

❑ **TurboVision Tools** 539
Un générateur d'applications TV sous TP 6, et une TPU avec source pour utiliser TV en EGA/VGA, un exemple très commenté.

❑ ❑ **Virus Survival Kit** 4080
Une disquette HD regroupant VScan 86, Virex 20, TBScan 32, et l'hypertexte sur les virus de P. Hoffman du 28/2/92.. Mise à jour dès parution des nouvelles versions.

❑ **Extended Batch Language** 1881
La version 4,07 apporte plus de 100 commandes supplémentaires au traitement par lot du DOS, pour le transformer en vrai langage de programmation.

Windows 3.0

❑ **Mind Frame 3.0** 1509
Système expert, capable d'intégrer textes et graphiques BMP dans des arbre de décision, et de générer des règles.

❑ **Clock manager Money smith** 1494
Gestion d'alarmes et d'évènements. Et Money Smith : comptabilité à double entrée avec une gestion de budget intégrée.

❑ **Jeux éducatifs Français** 1496
Logiformes, les hommes célèbres, logiphrases et le monde végétal.

❑ ❑ **Win 3.0 Icons** 1459,1474
Une collection de 1700 icônes et deux éditeurs pour en créer d'autres...

❑ **PixFolio 1.3** 1501
Pour visualiser, cataloguer et convertir des images aux formats bitmap usuels.

❑ **WinCheck 3.0 n** 1488
Gestion performante de compte bancaire et de finances personnelles.

❑ **Financier Plus** 1481
Gestion sophistiquée de (gros) patrimoine

❑ **WinFract 3.1** 1480
Pour tracer tout type de fractales par l'équipe du célèbre FractINT 16.11.

❑ **Visual Basic Tools** 1487
Utilitaires divers et code source.

❑ **Archives 3.08** 1484
Un (dé)compacteur de fichiers Lhz, Zip et Pak, un équivalent sous win 3.0 à Shez.

❑ **Reminder + Adress manager** 1489
Gestion d'adresses et d'emploi du temps, à la limite de la gestion de projets.

❑ **Win 3.0 Tools** 1493
Bizwiz : une vraie HP12C, CT shell : un gestionnaire de fichiers rapide et un gestionnaire de programmes plus puissant

❑ **vxBase for Visual Basic** 1503
Bibliothèque de fonctions, elle donne accès à la gestion de fichiers dBase / Clipper.

❑ **DP VB Tools & DDL** 1502
code source et exemples d'utilisation.

101 Utilitaires d'intérêt général pour Win 3.0 Un ensemble de 3 disquettes HD qui se décompacte sur 8 Mo de disque dur. Ref **W100** prix 150F

Graphismes

❑ **VGA animations** 1415h
4 belles animations Autocad, certaines en 3D. Il faut un disque dur.

❑ **Graphic Workshop 6.1 k** 1710
Pour visualiser, convertir, imprimer les images de tout format bitmap.

❑ ❑ ❑ **PrintMaster/Shop ART** 1422abc
bibliothèque de plus 2500 symboles & images (il faut PrintMaster ou PrintShop).

❑ **Print partner 1.1** 1706
Et Ligthning Press, clônes de PrintMaster.

❑ **PrintMaster Utilities II** 1705
Fait des étiquettes, convertit entre différents formats et imprime des chemins de fer de vos collections de clip art.

❑ **Image Gallery 1.0** 1862
Catalogue les collections d'images bitmap en établissant une base de données visuelles avec une miniature de l'image.

❑ **Desktop paint 2.3** 1707
Pour créer et retoucher des images bitmap (PCX, IMG, IFF, WPG, GIF, BMP.etc...) .

❑ **Recursive Realms 3.0** 1740
Logiciel de génération de fractales pour AT/VGA et disque dur. Bonne ergonomie.

❑ **Photos de Galaxies en VGA** 912
Très belles photos de galaxies, avec fiches descriptives des corps célestes.

Jeux CGA

❑ **Super-ZZT & Dark Ages** 865
Jeux d'arcade et d'aventures pour tout écran graphique.

❑ **Bridge 1.66** 841
Logiciel pour apprendre et progresser au bridge (en français).

❑ **Last Half of Darkness** 839
Beau jeu d'aventures graphiques (CGA à VGA) pour disque dur.

❑ **Robomaze II** 851
Jeu d'arcade et de réflexion, rapide, 40 niveaux, pour écrans CGA à VGA.

❑ **Paganitzu** (Apogee software) 863
Il mèle action et réflexion à la manière de Commander Queen, un futur "best of '91".

❑ **Jeux Othello's** 817
Compilés par un spécialiste de la Fédération Française. Pour grosses têtes.

Jeux EGA/VGA

❑ **La compil' Apogee pour AT** 829
Commander Keen, Monument of Mars, Pharao's Tomb : 3 beaux jeux d'arcade.

❑ **Tribolo, Sherlock, Battle Ship** 845
Une compilation de bons et beaux jeux de stratégie.

❑ **Duke Nukem & Boomerang** 850
Un clone de Mario's Brother, avec un bon graphisme et une animation fluide.

❑ **Jumpman Lives & Crusher** 846
2 jeux de reflexion et d'action écrits par les stars du genre : Apogee & .Soleau.

❑ **Crystal Cave & AntHill** 866
Nouveautés : jeux d'action et de réflexion.

❑ **Hugo II, Whodunit ?** 835
Aventures graphiques en 3 dimensions.

❑ ❑ **Megatron VGA 2.44** 871ab
Jeu de combat dans un labyrinthe 3D contre des robots (386 20Mhz et souris)

❑ **jeux de réflexion (a)** 856
Chinese Checkers, Bolo adventures, et Agression (il faut une souris).

❑ **jeux de réflexion (b)** 864
PC-Gammon IV (un superbe Back Gammon), Antic, Electronic Aggravation et Concentration 2.03 (Ega & souris)

Utilitaires

❑ **Shez 6.5, Lharc 2.13** 1308
Pour manipuler, éditer, convertir les fichiers compactés + lharc 2.13 probablement le meilleur des compacteurs récents.

❑ **PAK 2.51 et ARJ 2.30** 1065
2 Nouveaux compacteurs, plus puissants que ZIP, les successeurs à pkZip ?

❑ **Integrity Master 1.02** 1842
Surveille l'intégrité d'un disque dur et signale les détériorations, les attaques virales, les altérations de la FAT et du boot.

❑ **DOS 5 EA** 323
Explique point par point, les différences entre DOS 5 et les versions précédentes.

❑ **Anadisk 2.05 & Eddy 6.b** 1741
Le must en analyse de disque dur et disquettes. Plébiscité par nos utilisateurs.

❑ **Temps Menu 1.21** 1838
Gestion de menus et d'évènements, économiseur d'écran, éditeur, bloc-notes...

❑ **DOS 5 MemKit** 1799
Charge en mémoire haute les drivers sur les 8086 et 80286, optimise les 386.

❑ **Simply Labels 3.0** 1017
Pour gérer l'édition d'étiquettes sans soucis. Documentation en Français.

❑ **DOS for professionals** 1800
Extensions aux commandes DOS 3.0 à 5.0.

❑ **Stow Away 1.70** 1826
Libère de la place sur votre disque dur en déplaçant les fichiers peu utilisés sur des disquettes. Il maintient la liste des fichiers off line et en gère le rappel automatique.

Educatifs

❑ **Calculus Calculator** 975
Puissant calculateur scientifique avec programmation procédurale.

❑ **The Integral Scientist** 981
Base de données de physique-chimie et programmes utilitaires de calculs.

❑ **Mercury 2.06** 955
Un système complet pour résoudre les équations et faire le tracé de fonctions.

❑ **Geoclock 4.4 & World 2.99** 989
Horloge géographique qui montre en temps réel l'ensoleillment du globe (EGA/VGA), un atlas avec zoom (CGA).

❑ **Sky Globe 2.5** 900
Un des meilleurs programmes d'aide à l'observation astronomique (CGA à VGA).

❑ **Bota Logic** 1787
Une initiation à la botanique en français, idéal pour les élèves du primaire (CGA)

Programmation

❑ **Perfect Editor 2.82** 1839
un éditeur de programme moderne, avec des capacités traitement de textes.

❑ **Qedit 2.15** 1018
La dernière version du plus célèbre éditeur du monde Shareware.

❑ **RH-UCR Lib 2.4 for ASM** 312
Bibliothèque de routines d'intérêt général en ASM avec code source.

❑ **Dos Tech. Ref. Manual v 2.2** 108
540Ko de textes techniques indispensables sur tous les sujets, une bible!.

❑ ❑ **Interrupt List 2.92** 110ab
plus de 500 pages d'informations sur les interruptions DOS de 00 à FF, et autres.

❑ **Disassembler's ToolKit** 308
Utilitaires pour ceux qui s'intéressent au "reverse engineering".

❑ **C ++ Tutor 2.0** 115
Cours complet pour s'initier aux finesses du C++, il faut déjà connaître le C.

❑ **Turbo Pascal tutor 2.6** 512
Un didacticiel pour TPascal 6.0, approfondit la programmation orientée objet.

❑ **Snap ! 5.0** 746
Système documentaire pour programmes à la norme dBase, clipper et FoxBase.

❑ **CodeSmith** 751
Générateur d'applications clipper 5.01.

❑ ❑ **VP-Info 3.3** 729ab
Langage de base de données avec un compilateur, il lit les fichiers dBase et Clipper y compris les champs mémo.

❑ **C database Lib 1.02** 619
Fonctions de gestion de fichiers par séquentiel indexé, avec code source.

Bureautique

❑ **Intext** 1073
Traitement de textes multilingues (11 langues), dont le français, l'allemand, l'arabe.

❑ **Galaxy 1.7** 1012
Excellent traitement de textes pour tous et éditeur rapide pour programmeurs.

❑ **Instacalc 3.0** 1735
Tableur et grapheur, il peut lire et interroger des fichiers dBase (disque dur).

❑ ❑ **Gestion PME/PMI** HD 4094
Paye, facturation et gestion des immobilisations, des emprunts et crédits-bail, Il faut un lecteur HD et un disque dur.

❑ **ICP 7.0** 1882
Gestion d'inventaire, prend en charge les commandes, ventes et livraisons.Très Pro.

Bon de commande à retourner ou à faxer accompagné de votre réglement à : DP Tool Club, B.P. 745, 59657 Villeneuve d'Ascq

Nom:_____ Prénom:_____

Adresse:_____

Code Postal:_____ Ville:_____

❑ Demande le catalogue 900 disquettes (94 pages): joindre 5 timbres

❑ Commande des disquettes et le catalogue gratuit : ❑ 3"1/2 ❑ 5"1/4

_____ _____ _____ _____ _____ _____

_____ _____ _____ _____ _____ _____

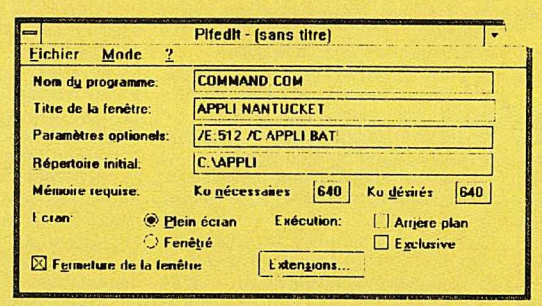
prix : 1D = **25F** en 5"1/4 et **27F** en 3"1/2 (❑ =1D, ❑ ❑ =2D,❑❑❑ =3D)

Pas de commande minimum ! Port 16 F (France, C.E.E.,Suisse)

Une facture avec TVA est fournie avec chaque commande.

Paiement par Carte Bleue/MC: indiquez le N° la date d'expiration et signez:

FIGURE 1



mètre /C suivi d'un espace puis du nom du batch) du fichier batch déjà présenté plus haut.

Le piège de l'EMS

Au chapitre de la mémoire EMS, nous faisons également des découvertes d'importance et nous y trouvons une difficulté au moins aussi vicieuse que la précédente.

WINDOWS en mode 386 étendu présente la particularité d'offrir une émulation EMS à toutes les boîtes DOS et par contrecoup aux applications DOS fonctionnant à l'intérieur de ces boîtes DOS. Seulement, il peut arriver qu'il n'y ait pas de disponibilité suffisante de mémoire dans la plage comprise 640 KO et 1 MO pour l'implantation d'une fenêtre EMS.

Dans ce cas, vous pourrez créer autant de fichiers PIF que vous le désirez en demandant la mémoire EMS (page écran "Extensions" dans l'utilitaire PIFEDIT), vous n'en obtiendrez pas et votre bon vieux LOTUS en version 2 restera désespérément coincé à la taille de la mémoire conventionnelle disponible dans la boîte DOS, plage de mémoire qui rétrécit comme une peau de chagrin.

53, rue Corot · La Rochette · 77000 MELUN · FRANCE · Tél: 33(1)64 39 13 33 · Téléfax: 33(1)64 39 17 81

ARGO.S

UN NOUVEAU CONCEPT DANS LE DOMAINE DE LA PROTECTION DE LOGICIELS

- Notre expérience dans le domaine de la protection, notre pouvoir d'innovation, nous permettent de vous proposer le seul produit Français utilisant des composants du type ASIC, pour une clé gigogne.
- Avantages : Circuit intégré haute densité exclusivement fabriqué pour Electryon, permettant une clé de dimension réduite, une fiabilité au fonctionnement, une transparence parfaite et une protection absolue.
- Plus de 50 fonctions vous permettent d'exploiter l'ensemble des ressources de la clé.
- Traitement sur 16, 8, 1 bits, compteur, dates.
- Des registres spécifiques 32 bits : code éditeur, numéro de série (public), numéro d'utilisateur final, code SSII (Software house).
- Des mots de passe définis par l'éditeur.
- Protection de compilateurs, générateurs, utilitaires et des logiciels générés par ces derniers.
- Possibilité d'interfaçage avec tous les langages de programmation sur PC.
- Clés et logiciels personnalisés.

ENCORE PLUS PERFORMANTE

- Protection de 20 sociétés parmi 4000. Chacune de ces sociétés peut protéger plusieurs logiciels
- Activation et désactivation de la clé par programme.
- Garantie : de 1 à 3 ans selon modèle.

DEMANDEZ NOTRE DOCUMENTATION

ELECTRYON

Arrêtez de développer ce qui existe déjà...

COMPILEZ... C'EST GAGNÉ !

FINI

La programmation sans fin
Les séances de debugging
Les essais interminables

AGENDA Multi-Utilisateur

Cette gestion des rendez-vous très visuelle, vous sera très vite indispensable. Elle gère l'utilisation en multi-poste, permet la programmation des rendez-vous, de rechercher des plages de temps libres. Possibilité de couper-coller des rendez-vous.

- Version exécutable bridée..... **150 F H.T.**
- Version exécutable..... **649 F H.T.**
- Source en TP 6.0..... **2490 F H.T.**

CODE A BARRE

Impression des codes à barres au standard EAN 13, 2/5 entrelacé et Alpha 39.

- Version exécutable..... **649 F H.T.**
- Source en TP 5.5..... **1990 F H.T.**

GESTION DES STOCKS FACTURATION - CAISSE

Gestion des fichiers produits, clients, fournisseurs. Permet d'éditer vos devis, factures, avoirs, bons de commande, bons de livraison. Gère les ventes au "comptoir", édition des tickets de caisse, du ticket récapitulatif. La version Stock 4 comporte en plus une gestion du multiposte, une aide contextuelle, une gestion des représentants, les taxes parafiscales, un générateur d'état, des éditions paramétrables, des sélections multi-critères pour toutes les éditions (listes, courriers, fiches récapitulatives, etc...), import-export de fichiers en ASCII.

- Version exécutable de Stock 4 bridée **150 F H.T.**
- Version exécutable de Stock 3..... **430 F H.T.**
- Version exécutable de Stock 4..... **1850 F H.T.**
- Source Stock 3 en TP 3.0..... **2990 F H.T.**
- Source Stock 4 en TP 5.5..... **5990 F H.T.**

GESTION FICHIER COMMERCIAL

Gestion d'un fichier de prospection. Grâce à son générateur d'état intégré, il vous offre la possibilité de mailing ciblé et personnalisé et d'édition d'étiquettes. Le programme gère le multiposte, comporte une aide contextuelle, etc...

- Version exécutable bridée..... **150 F H.T.**
- Version exécutable..... **1260 F H.T.**
- Source en TP 5.5..... **3990 F H.T.**

SIMASOFT le concepteur et l'éditeur de logiciels le plus souple du marché, vous permet désormais d'acquérir les sources de plusieurs de ses programmes (écrits en Turbo Pascal). Vous aurez alors la permission de revendre vos nouvelles versions compilées sans limitations quantitatives et sans payer de redevances. Pour ceux qui ne possèdent pas le Turbo Pascal nous pouvons aussi vous personnaliser ces logiciels.

Nos logiciels sont utilisés par milliers dans de nombreux domaines (PME-PMI, artisans, commerçants, professions libérales, médicales, para-médicales), d'où une garantie de fonctionnement irréprochable.

Si vous ne connaissez pas nos logiciels, des versions bridées (déductible d'un achat ultérieur) vous permettront de juger de leurs qualités.

Nos logiciels fonctionnent sous MS-DOS (Version = 3.0) sur un ordinateur compatible PC avec 640 K de RAM et un disque dur. Pour compiler les sources, vous devez posséder le compilateur Turbo Pascal de Borland.

Nos prix sont HT (TVA 18.6%). Une facture justificative est jointe aux envois. Les commandes doivent impérativement être accompagnées du règlement du montant total (logiciel + frais de port) soit par chèque, soit par mandat. Les frais de port sont de 29 F par commande, ou de 50 F par envoi en contre-remboursement, ou de 100 F pour les DOM-TOM et l'exportation.

Rédigez votre bon de commande de manière lisible, en indiquant vos nom, prénoms, adresse complète, Numéro de téléphone, ainsi que le format des disquettes que vous utilisez. Datez, signez et envoyez votre commande à l'adresse ci-dessous :

SIMA SOFTWARE

12, RUE DE LA RÉPUBLIQUE
13420 GÉMENOS - TÉL. 42.32.09.32
FAX. : 42.72.66.54 ET 42.32.17.27

COMPTABILITE GENERALE

Comptabilité générale multi-sociétés. Saisie des écritures par brouillard. Lettrage automatique et manuel. Editions des brouillards, journaux, grands-livres, balances, bilan, compte de résultat. La version II gère en plus le multiposte, des guides de saisies automatiques.

- Version exécutable bridée PROFIMAX II **150 F H.T.**
- Version exécutable PROFIMAX I..... **449 F H.T.**
- Version exécutable PROFIMAX II..... **1690 F H.T.**
- Source PROFIMAX I en TP 3.0..... **3490 F H.T.**
- Source PROFIMAX II en TP 5.5..... **5990 F H.T.**

COMPTABILITE LIBERALE 2035

Comptabilité simplifiée pour les professions libérales (soumises ou non à la TVA). Saisie en mode recettes - dépenses.

- Version exécutable bridée..... **150 F H.T.**
- Version exécutable..... **1690 F H.T.**
- Source en TP 6.0..... **5990 F H.T.**

GESTION DE LA PAYE ET DU PERSONNEL

Paye multi-sociétés entièrement paramétrable. Calcul de la paye mensuelle, horaire, paye à l'envers. Editions des bulletins aux nouvelles normes. Gestion des congés payés, des repos compensateurs, calcul de la CSG. Editions du livre de paye, des fiches individuelles, des fiches récapitulatives, de l'état préparatoire à la DADS, des avis de virements. Gestion des règlements, des acomptes.

- Version exécutable bridée..... **150 F H.T.**
- Version exécutable..... **1790 F H.T.**
- Source en TP 5.5..... **5990 F H.T.**

SENATOR II
GESTION COMPLETE D'ENTREPRISE

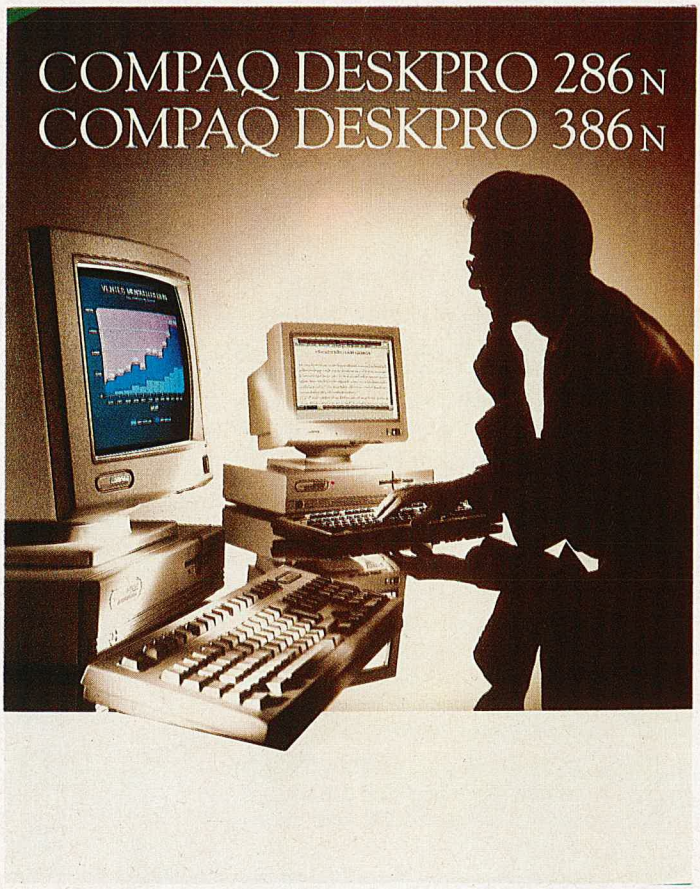
Reprend toutes les fonctionnalités du programme Stock 3 (GESTION DES STOCKS - FACTURATION - CAISSE), du programme Profimax I (COMPTABILITE GENERALE). De plus il contient un traitement de texte et une gestion de fichier commercial. Le tout est lié dans un ensemble cohérent et interactif (Transfert des factures, avoirs en comptabilité, transfert des prospect en gestion de stock, accès au traitement de texte pour toutes les éditions, Etc...

- Version exécutable..... **1260 F H.T.**
- Source en TP 3.0..... **6990 F H.T.**

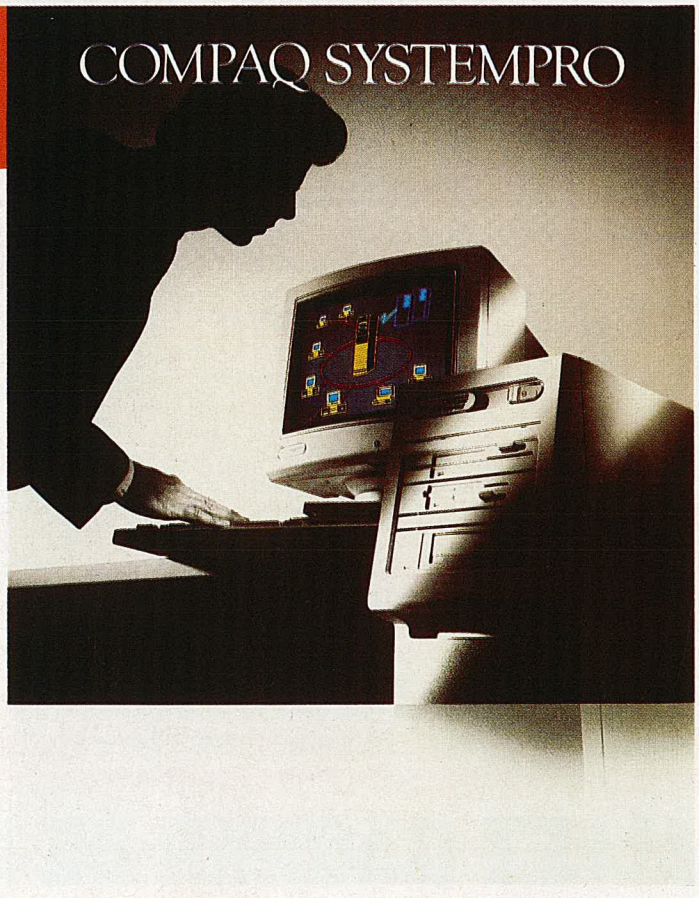
VOUS ALLEZ GAGNER DE L'ARGENT

PAS DE REDEVANCES

Equipés respectivement des microprocesseurs 286 à 12 MHz et 386SX à 16 MHz, le **COMPAQ DESKPRO 286N** et le **COMPAQ DESKPRO 386N** offrent en standard toutes les fonctionnalités des micro-ordinateurs professionnels : 3 connecteurs d'extension (2 connecteurs 8/16 bits ISA et 1 connecteur dédié à la mémoire vive rapide), 1 Mo de mémoire vive extensible à 13 Mo (286N) et 16 Mo (386N), jusqu'à 120 Mo de mémoire de masse en interne, 2 emplacements de mémoire de masse, un nouveau contrôleur graphique VGA accéléré avec support du mode texte 132 colonnes.

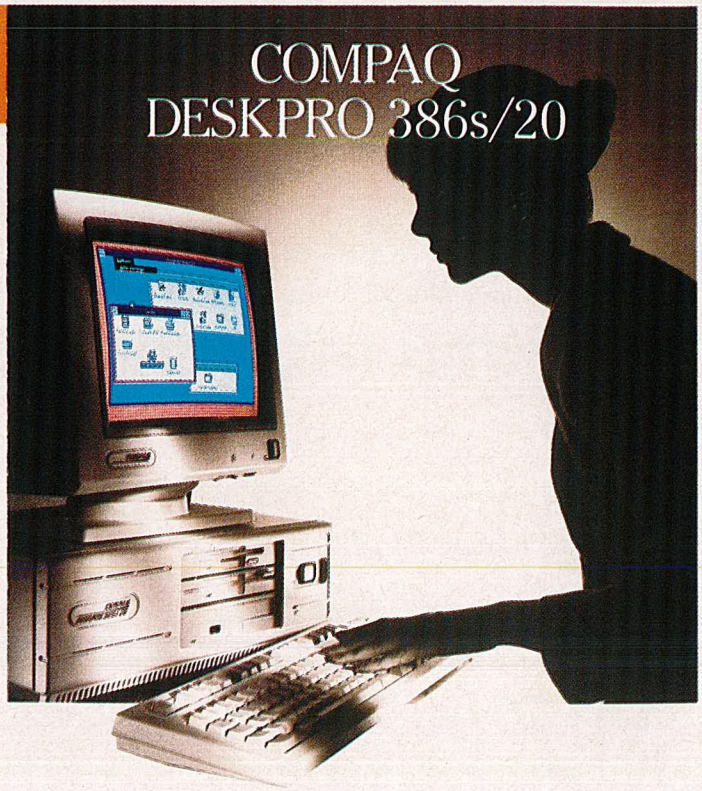


COMPAQ DESKPRO 286 N
COMPAQ DESKPRO 386 N



Basé sur une architecture évolutive en termes de processeur et de mémoire de masse, le **COMPAQ SYSTEMPRO** a été le premier micro-ordinateur à offrir un mode multi-processeur et un système multidisque. Il évolue ainsi avec les besoins des utilisateurs et préserve leurs investissements. Cette nouvelle génération d'ordinateurs a connu un grand succès : de nombreuses entreprises sont en train d'adopter le **COMPAQ SYSTEMPRO** pour faire évoluer leurs sites centraux et leur mini-informatique vers une informatique ouverte et au standard de l'industrie.

COMPAQ SYSTEMPRO



Equipé du dernier microprocesseur INTEL 80386SX à 20 MHz, le **COMPAQ DESKPRO 386s/20** est disponible en 3 modèles : 1,60 et 120 (disques durs avec temps d'accès inférieur à 19 ms). Outre sa mémoire cache ultra-rapide, le **COMPAQ DESKPRO 386s/20** est doté de nombreuses innovations. Son contrôleur VGA est ainsi accéléré pour offrir un maximum de performances en environnement graphique. De plus, il supporte le mode texte 132 colonnes utilisé par un nombre grandissant de logiciels de type tableur, traitement de texte ou d'émulation terminal.

COMPAQ DESKPRO 386s/20

Le **COMPAQ DESKPRO 486/33L** est aujourd'hui le micro-ordinateur de bureau le plus puissant du monde. En plus d'un coprocesseur 80387 et de 8 Ko de mémoire cache intégrés au processeur i486, sa carte processeur comprend un cache de deuxième niveau de 128 Ko de mémoire ultra-rapide. Ce nouveau micro-ordinateur est conçu pour des applications spécialisées techniques telles que la CAO, le développement logiciel, les SGBD et les applications scientifiques.

Le **COMPAQ DESKPRO 386/33L** dispose d'une carte processeur qui est équipée du microprocesseur 386 à 33 MHz et de 64 Ko de mémoire cache. Il constitue l'outil idéal pour les applications spécialisées telles que la CAO, les SGBD, la modélisation et l'analyse financière, ainsi que la PAO.
Un programme d'échange de la carte processeur 386 contre la carte i486 va permettre d'assurer aux utilisateurs la pérennité de leurs investissements, tout en répondant à leurs besoins d'évolutivité.

LA DEDICACE INFORMATIQUE

COMPAQ
REVENDEUR AGREE

173, rue de Vaugirard 75015 PARIS
Tél. 45.66.94.69
Fax : 43.06.35.18 - Télex 202 993

COMPAQ DESKPRO 486/33L
COMPAQ DESKPRO 386/33L

Création Parenthèse - Paris - 43 55 76 20

Le plus difficile à admettre, c'est que, dans l'immense majorité des cas, l'espace nécessaire à l'implantation de la frame EMS est disponible, mais sous forme de morceaux épars. On en revient toujours à la même chose : la configuration matérielle d'une machine est une affaire de professionnels.

Selon le bon vieux principe des vases communiquants que l'on étudie en primaire, ce que vous ne pouvez pas mettre en haut va forcément en bas, et à l'époque de la communication, il n'est plus rare de bloquer 120 KO de mémoire conventionnelle pour le chargement des couches logicielles nécessaires à l'accès à un réseau local qualifié d'hétérogène. Dans des configurations un peu musclées, on trouve rapidement un mélange des couches IPX (NOVELL), TCPIP (UNIX) et autres NETBIOS toutes aussi indispensables les unes que les autres à l'échange d'informations à l'intérieur des entreprises.

Comment envisager alors de travailler sereinement avec des applications DOS elles aussi de plus en plus gourmandes en mémoire sans essayer de repousser les limites de notre cher vieux système d'exploitation ?

Le vrai contexte WINDOWS 3.0

A l'évidence WINDOWS est une réponse du fait de sa capacité à gérer plusieurs boîtes DOS simultanément.

A l'évidence, DOS 5.00 complète l'apport de WINDOWS en libérant au moins 40 KO de plus de mémoire conventionnelle que ses prédécesseurs.

A l'évidence enfin, un produit comme QEMM 386 version 6.0 permet de parfaire l'ensemble.

J'ai lu récemment dans un magazine un test sur QEMM 6.0. A ma grande stupeur, j'y ai découvert que les testeurs des produits n'avaient récupéré que 96 malheureux kilos octets de mémoire.

De 2 choses l'une : ou bien ils n'ont pas su utiliser QEMM, ou bien ils n'en avaient pas réellement besoin !

Si, aujourd'hui sur le marché, il existe un produit qui dépasse de très loin tout ce que nous connaissons de la gestion de la mémoire sous DOS, c'est bien QEMM 6.0 !

J'ai installé QEMM 386 sur des PS/2 en réseau local et j'ai récupéré, en plus d'une fenêtre EMS de 64 KO, 192 KO de mémoire, c'est-à-dire assez pour les couches réseau, toutes celles mentionnées plus haut.

La même opération sur des clones Taïwanais nantis d'un BIOS d'origine PHOENIX m'a donné exactement les mêmes résultats !

Pour compliquer les choses, les PS/2 étaient amputés d'une partie de leur mémoire haute du fait de la présence de :

- une carte TOKEN RING (16 KO de ROM + 16 KO de RAM),
- une carte vidéo 8514/A (4 KO),
- une carte d'accès aux sites centraux (connexion 3270) pour 8 KO,
- un BIOS occupant 128 KO d'origine (Merci IBM !).

Pour couronner le tout, l'ensemble de la configuration — pour le moins compliquée — fonctionne parfaitement sous WINDOWS, y compris les applications DOS que j'ai désormais équipées d'un superbe PIF tout de COMMAND.COM paré.

Même dopé de la sorte, il n'en reste pas moins vrai que WINDOWS 3.0 souffre d'un certain nombre de limitations, la plus cruelle étant à mon avis la trop faible disponibilité de "Ressources Système" de sorte qu'il ne soit désormais pas rare de tomber en panne de ce type de ressources avant même d'avoir épuisé les ressources mémoire, elles, largement boostées par le phénomène de Paging du mode 386 étendu.

Malheureusement pour nous, la toute prochaine version qui porte déjà le numéro 3.1, ne corrige que très partiellement cet handicap.

De sérieuses améliorations sont cependant à porter à son actif :

- en terme de performances tout d'abord : MICROSOFT annonce 30 à 40 % de mieux !
- en terme de fonctionnalités ensuite avec, par exemple, les polices TRUE TYPE qui vont notablement amputer le marché d'ADOBE TYPE MANAGER et autres SUPER PRINT ou FACE LIST.

Bulletin d'abonnement pour 1 an

10 numéros au prix de : France : 220 F T.T.C., Etranger : 300 F
Je souscris un abonnement d'un an à la Revue de L'utilisateur de l'IBM-PC

Nom : .....
Société : .....
Adresse : .....
.....
Code postal : ..... Ville : ..... Tél. : .....

Chèque bancaire joint à l'ordre de SOFtin Communications (France).

A renvoyer à : La Revue de l'Utilisateur de l'IBM—PC, 2, rue d'Amsterdam, 75009 Paris.

Avec 5 millions de copies vendues, MICROSOFT peut envisager sereinement les années 90 : le monde des microprocesseurs INTEL va désormais fonctionner sous WINDOWS.

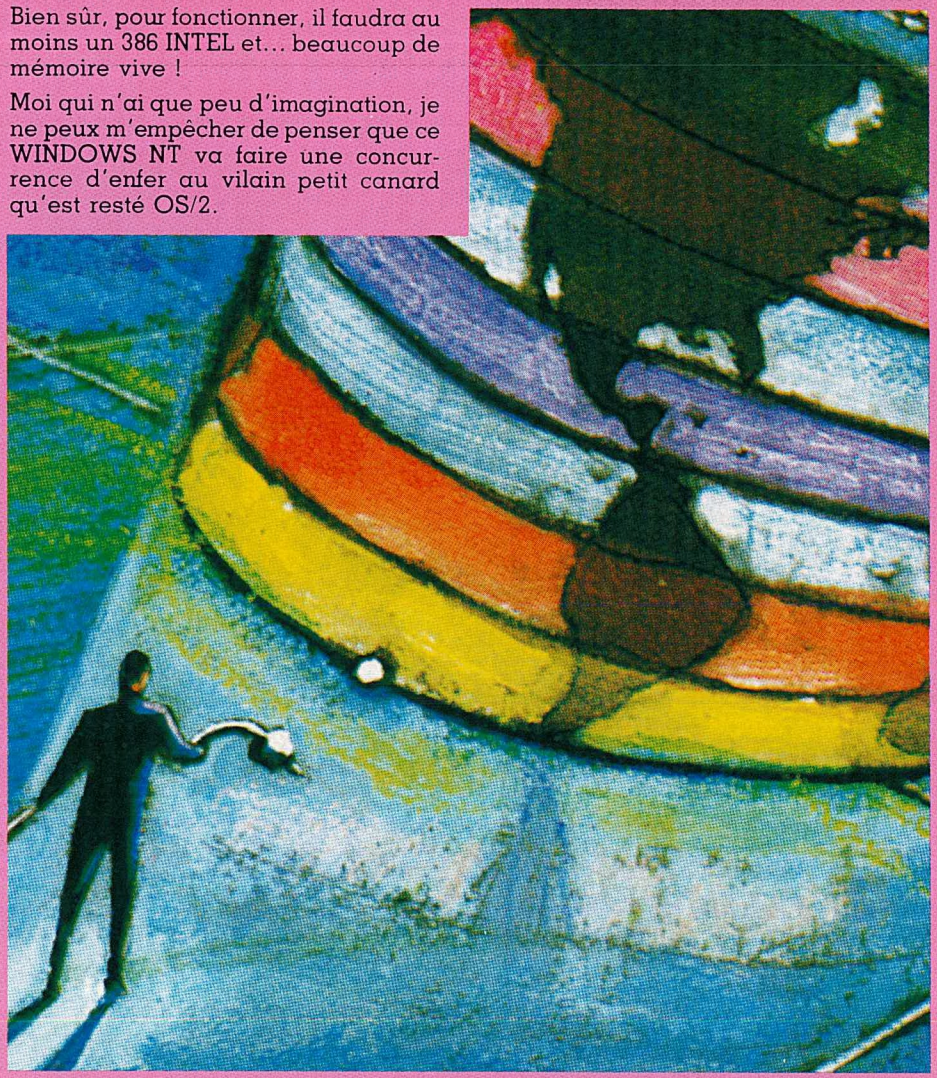
Charles PETZOLD, l'auteur du mythique ''Programmer sous WINDOWS'' désormais intégré au package français du QUICK C pour WINDOWS s'était légèrement trompé quand il écrivait : ''... il est à peu près sûr que WINDOWS ne sera pas porté sur d'autres environnements...''.

Dans moins d'un an, WINDOWS sera disponible en version NT (NT veut dire New Technology ou Nouvelle Technologie pour les acharnés du Larousse), sur des machines aussi diverses que les stations DIGITAL, les machines RISC ou encore les machines de MIPS Computer qui développeront la bagatelle de 65 MIPS, de quoi faire rêver le ''heureux propriétaire'' de 486 à 33 MHZ qui rament à 15 ou 17 MIPS.

WINDOWS NT est un système d'exploitation à part entière (par exemple, il ne délégue pas les accès disques à la couche DOS) capable de faire tourner les applications WINDOWS 16 bits — celui que nous connaissons aujourd'hui — mais aussi les nouvelles applications WINDOWS 32 bits, avec une gestion préemptive de la répartition du temps machine — un process ne peut pas pas bloquer indéfiniment l'ensemble des ressources microprocesseur pour son usage exclusif —, sans compter les applications DOS traditionnelles.

Bien sûr, pour fonctionner, il faudra au moins un 386 INTEL et... beaucoup de mémoire vive !

Moi qui n'ai que peu d'imagination, je ne peux m'empêcher de penser que ce WINDOWS NT va faire une concurrence d'enfer au vilain petit canard qu'est resté OS/2.



COUIC

L'outil à désassembler

Que vous **chassiez le virus** ou que vous cherchiez à comprendre comment fonctionne un programme **COM** ou **EXE**, que vous soyez programmeur professionnel ou débutant, il vous faut un bon **désassembleur immédiat**, véritable **fenêtre ouverte** sur le code machine.

— Vous devez pouvoir **désassembler un programme** en ROM, en RAM sur disque ou sur disquette en spécifiant une adresse, **un numéro d'interruption**, un nom de fichier ou un **secteur** et faire défiler le listing de désassemblage ou de dumping en avant et en arrière.

— Vous devez disposer d'instructions **puissantes** de localisation automatique d'**interruption**, de **suivi de jump**, ou de **recherches** de chaînes et de données.

— Vous devez disposer de commandes suffisamment simples pour être apprises et manipulées sans difficulté mais assez efficaces pour faciliter la lecture du code et effectuer des **marquages** dans les zones de données.

— Vous devez pouvoir générer des **fichiers séquentiels** utilisés par les désassembleurs symboliques.

— Vous devez disposer du **code source commenté** de votre désassembleur de façon à l'adapter à vos besoins spécifiques.

COUIC-DESAS permet tout cela et plus encore grâce à ses fonctions spéciales et ses tables descriptives d'interruptions.

Exceptionnellement COUIC est vendu avec ses sources en langage "C" : plus de **9 000 lignes** de code réutilisable dans vos applications et sa documentation pour seulement **340 F (TTC)**.

COUPON
à adresser à La Revue de l'Utilisateur de l'IBM PC
2, rue d'Amsterdam
75009 Paris
Tél. : 47.64.07.57. - 42.67.93.80.

Nom.....

Adresse

.....

Code Postal.....**Ville**

Téléphone.....

Veuillez m'envoyer ..... **logiciel(s) COUIC au prix unitaire de 340 F (TTC).**

Ci-joint un chèque de ..... **F. En règlement à Softin Communications.**

Le traitement d'image Mise au point d'un outil général de traitement et de manipulation d'images (1re partie)

Jean-Michel SPINER

Modifier la luminance d'une image, la réduire d'un facteur non entier, y appliquer des filtres de couleurs : ce sont quelques-unes des fonctions du système que nous allons aujourd'hui mettre en place.

Le but est de créer un outil général, qui va permettre d'appliquer à une image au format PCX tout un éventail de traitements choisis par le programmeur.

Ces traitements pourront se faire sur la couleur, sur la forme, ou sur la taille de l'image.

Structure du système

Conçu pour être utilisé en interactif ou en automatique, le programme boucle sur l'attente d'une ligne de commande, qui est ensuite interprétée.

La première chose à faire est de charger l'image de départ.

Ce chargement s'effectue grâce à la commande l (load) avec la syntaxe l ⟨nom⟩ ⟨num⟩ ; nom désigne le nom du fichier PCX à charger, et num le numéro du buffer dans lequel sera rangé cette image.

En effet, une des règles du traitement d'image est qu'il faut, dans la mesure du possible, travailler en mémoire centrale, et non en mémoire vidéo.

Cette façon de procéder offre deux avantages : d'une part, il est ainsi possible de stocker plusieurs images et d'effectuer des traitements de type fusion, ou addition, et d'autre part, les temps de traitement sont de loin inférieurs.

Une image pourra ainsi prendre place dans le système en fonction de la mémoire non allouée restante.

Sans mémoire étendue, on pourra travailler simultanément sur 8 ou 9 images de taille maximale en mode 320 × 200 en 256 couleurs (VGA 13h).

L'image va prendre place dans une structure définie ainsi :

x1 et y1 désignent l'origine de l'image, w et h sa taille.

La variable in_use est à TRUE si le buffer est utilisé, à FALSE dans le cas inverse.

ptr_data pointe sur la zone mémoire allouée à l'image. En mode 256 couleurs, un octet représente un pixel, la taille de la zone est donc de w*h, à un bloc près.

ptr_pal contient la table de couleur c'est-à-dire 256 fois le triplet RVB qui relie la valeur d'un pixel, sur un octet, à sa couleur effective, qui est codée sur trois octets.

Si vous êtes déjà un utilisateur de CLIPPER de Nantucket, vous avez besoin du

Générateur de programme
ATELIER

Il fait pour vous :

☐ la génération du menu principal,
☐ la génération de sous-menus,
☐ la génération des programmes de saisie,
☐ la génération des programmes d'impression,
☐ la gestion de vos sources,
☐ le lancement du compilateur...

ATELIER

☐ incorpore votre copyright et une calculette dans vos programmes
☐ génère le code fastidieux (90 %)
☐ et vous laisse faire la partie intéressante de la programmation.

ATELIER

☐ est livré avec les sources commentées, ainsi qu'une documentation imprimable.

La version compilée n'est pas livrée sur la disquette car ATELIER est destiné aux utilisateurs de CLIPPER et nécessite ce compilateur.

Les commentaires sur ce programme ont été publiés dans La Revue de l'Utilisateur de l'IBM PC (numéro 59, Janvier-Février 1990), et la suite de la série comportera 6 articles à compter du numéro 61, Avril 1990.

ATELIER

est disponible auprès de La Revue de l'Utilsateur de l'IBM PC au prix de 450 F, sur disquette 5''1/4 ou 3''1/2.

COUPON
à adresser à La Revue de l'Utilisateur de l'IBM PC
2, rue d'Amsterdam
75009 Paris
Tél. : 47.64.07.57. - 42.67.93.80

Nom ..... **Prénom** .....
Adresse .....
.....
Code Postal ..... **Ville** .....
Téléphone .....
Veuillez m'envoyer ..... logiciel (s) ATELIER au prix unitaire de 450 F (TTC).
Veuillez trouver ci-joint un chèque de ..... F en règlement à Softin Communications.

```
typedef struct struct_buffer {
   unsigned short x1;
   unsigned short y1;
   unsigned short w;
   unsigned short h;
   char in_use;
   unsigned char huge *ptr_data;
   struct struct_coul ptr_pal[MAX_COLOR];
};
```

La libération d'un buffer est obtenu par la commande s (supress).

Enfin, une fois le traitement effectué, la commande w (write) va permettre de transformer le contenu du buffer en un fichier PCX qui pourra relire votre palette favorite.

Pour ceux qui ne possédaient pas La Revue de l'Utilisateur PC n° 72, (Septembre 1991), consacrée au format PCX, nous allons faire un petit rappel.

Mise au point par la Société ZSoft's, il est reconnu par la grande majorité des applications graphiques.

Les 128 premiers octets d'un fichier PCX constituent un en-tête qui contient entre autres la longueur et la hauteur de l'image, le nombre de plans, et de bits par pixels.

En mode VGA 13h, cet en-tête ne peut contenir la table de couleurs. Celle-ci est donc reléguée à la fin du fichier.

A l'offset 128 du fichier vient la zone de données, pour laquelle les fichiers PCX utilisent un codage appelé RLC (Run Length Coding).

Toutes les fonctions qui permettent de charger, de sauver, ou d'extraire la palette d'un fichier PCX sont regroupées dans le listing PCX13h.c.

Voilà défini le corps du système. Nous pouvons à présent attaquer la partie traitement.

Les commandes proposées

— display

Permet de visualiser le contenu d'un buffer. L'affichage passe en mode 13h, et l'image est présente jusqu'à la frappe d'une touche.

Il faut cependant noter qu'un programme de traitement d'image peut très bien travailler dans des résolutions qu'il est incapable d'afficher, le fichier résultant pouvant être destiné à l'affichage sur une autre machine.

— filter

Nous avons ici développé un exemple de filtrage de couleurs, qui permet d'augmenter ou de diminuer les valeurs affectées aux composantes, soit en rouge, vert, bleu, soit, ce qui est plus intuitif, en teinte, saturation, luminance.

Pour éclaircir une image, par exemple, on lui appliquera un filtre de type luminance avec une valeur de 20.

Pour l'affadir, on utilisera un filtre en saturation de valeur – 30.

Ces traitements ont la particularité de s'effectuer, non sur la zone de données, mais sur la table de couleurs.

Pour passer du rvb ou tsl, et inversement, on utilise les fonctions de conversion Rvb__tsl (r,v,b,&t,&s,&l) et Tsl__rvb (t,s,l,&r,&v,&b) qui se trouvent dans TRAITE.C.

— help

La fonction d'aide affiche la liste et la syntaxe des commandes disponibles.

— mode

Certaines cartes VGA, parmi les plus anciennes, ne supportent pas l'adressage mémoire direct.

Mode permet ainsi de passer par le BIOS lors de l'affichage par la commande display.

Les disquettes de la Revue

ARTOOL UN LOGICIEL DE D.A.O. POUR PC ET CARTES EGA OU VGA

ARTOOL possède toutes les fonctions pour exploiter pleinement les possibilités graphiques de votre PC muni d'une carte EGA ou VGA. Il fonctionne en mode BITMAP, et gère toute la palette de couleurs.

Spécifications techniques :

☐ Fonctionnement en 16 couleurs, 640×350 pour EGA et 640×480 pour VGA.

☐ Dessin à "main levée", tracé de droites, de cercles et de boîtes.

☐ Ajustement d'une courbe par la méthode de Bézier .

☐ Fonctions avancées : loupe, miroir, coloriage, texte.

☐ Gestion de la palette, des fichiers images et de l'imprimante.

ARTOOL fonctionne sur tous compatibles PC muni d'une souris et d'une carte graphique EGA ou VGA. Un minimum de 640 Ko ainsi qu'un disque dur sont vivement recommandés. Des impressions graphiques peuvent être obtenues sur des imprimantes compatibles IBM-GRAPH ou EPSON.

Fourni avec les sources en langage PASCAL, vous pourrez adapter ARTOOL à vos besoins et à votre configuration.

ARTOOL est disponible sur disquette 5'' 1/4 (360 Ko), au prix de 450 F (TTC)

BON DE COMMANDE
à adresser à La Revue de l'Utilisateur PC
2 rue d'Amsterdam
75009 Paris - Tél. : 42.81.54.27

Nom .....

Adresse .....

Code Postal ..... Ville ..... Téléphone .....

Veuillez m'envoyer ..... logiciel(s) ARTOOL, au prix unitaire de 450 F TTC.

Ci-joint un chèque de ..... F.

En règlement à l'ordre de Softin Communications.

LDQS LE VRAI SPECIALISTE CLIPPER

Avec **Class(y)**, Clipper 5.01, au même titre que C++, dès maintenant est un langage pleinement orienté objet, pleinement testé et mis au point, et disposant de bibliothèques de classes indépendantes.
Il permet donc des modules beaucoup plus simples à réutiliser et à adapter.(1490 Frs h.t.)
En même temps, Class(y) ne vous oblige pas à foncer dans la programmation objet tête baissée.

StarClass est une bibliothèque de gestion d'interface-usager, construite au moyen de Class(y). Elle permet la gestion de tâches en arrière-plan simulé, la gestion d'évènements par touche-clef (hotkey), et surtout la gestion de fenêtres virtuelles, mobiles à volonté, comprimables, à déroulement, etc.
Ces fenêtres virtuelles permettent toute forme de Get ou de Browse, sur Array , Memo, Fichier
StarClass est complètement utilisable tel que livré, sans Class(y).(1590 Frs h.t.)

ClipSQL donne à vos applications Clipper un accès complet à SQL Server sous OS/2 et à Sybase sous Unix et sur DEC VAX. ClipSQL fait de Clipper le langage optimal pour le développement d'applications client-serveur, avec opération en arrière-plan véritable,accès simultané à des bases de données multiples.(3600 F h.t.)

FasTBrowse Un générateur d'objets TBrowse.
Vous générez des browses multi-fichiers en couleurs et avec des fonctions spécifiques, des écrans multi-browses sur mesure, et toute fantaisie que l'on puisse avoir envie de réaliser avec des TBrowse.(890 Frs h.t.)

Demandez notre catalogue complet avec explications techniques

SEMINAIRE LDQS CLIPPER 5 de NANTUCKET A PARIS
SEMINAIRE NIVEAU AVANCE 5.01

19-20 MAI

Le développement efficace en Clipper 5: Environnements de travail; programmation dynamique; gros projets.
Programation **O**rientée **O**bjet (POO); réutilisation réelle; traduction ; besoins-conception-code.

21-22 MAI

Les applications commerciales et industrielles en Clipper 5:
Conception; gestion de l'interface; fichiers multi-formats. Gestion sur réseaux; sécurité; stabilité.

Totalement axés sur la pratique, truffés d'exemples précis, les séminaires LDQS vous donnent des solutions concrètes pour vos applications.
Chez LDQS,la théorie est seulement un moyen pour résoudre des problèmes réels et produire des résultats.

Principaux intervenants
GREG LIEF ,auteur de GRUMPFISH,**IAN HILL** auteur de NOVLIB,**PHILIPPE RANGER** spécialiste POO
Agrément formation continue.Repas midi inclus.Nombre de participant limité.
Participation: 3800H.T.Frs par session pour inscription avant le 21 Avril.(+500 Frs après cette date)

Attention ! Vous n'avez pas de temps à perdre. Ce séminaire est un séminaire de travail directement orienté vers la productivité.Nous vous garantissons que tous les outils présentés sont disponibles actuellement .
Nous vous épargnerons l'éventuel CLIPPER pour Windows à paraitre fin 93.

LDQS - 5, Rue Eugène Emanuel - 06000 NICE - TEL. 93 16 09 12 - FAX . 93 88 08 90

LDQS - 5, Rue Eugène Emanuel - 06000 NICE - TEL. 93 16 09 12 - FAX . 93 88 08 90

BULLETIN D'INSCRIPTION DU SEMINAIRE 5.01

Session 1 le 19-20 Mai ☐ Session 2 le 21-22 MAI ☐

Nom : Société

Adresse : Code Postal Ville

TEL : FAX :

Nombre de participants : CI-joint un chèque de : Frs

— quit

Fin du programme.

— resize

S'il est simple de diminuer une image de moitié, en ne considérant qu'un pixel sur deux, il faut utiliser un algorithme un peu plus complexe pour passer d'une taille quelconque à une autre.

La méthode est décrite par les fonctions Resize, Makexmap et Xscalebuf du listing TRAITE.C. Elle est appelée méthode du zoom à impulsion, et est relativement performante.

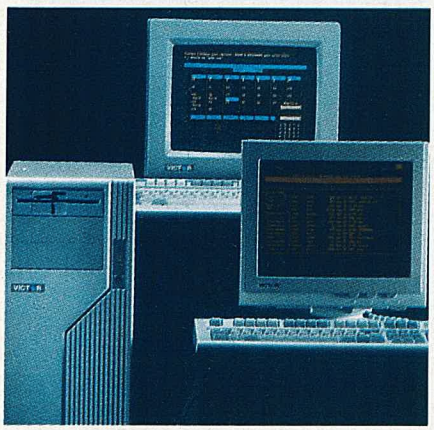
Bien entendu, tout zoom entraîne une perte d'informations donc une dégradation de la qualité visuelle. La limitation à 256 couleurs nous interdit de tempérer cette perte à l'aide de l'anti-aliasing sans affecter la table de couleurs.

Développements futurs

Nous allons pour conclure jeter en vrac quelques idées de traitements. Vous en aurez certainement beaucoup d'autres, n'hésitez pas à nous envoyer vos réalisations.

— Fonctions de floppage : effet miroir ou effet tête en bas.

— Fonctions de rotation : 90, 180, ou 270 degrés dans un premier temps, angle quelconque ensuite.



— Cryptage et décryptage.

— Conversion en noir et blanc avec seuillage.

— Conversion en monochrome.

— Suppression des points isolés.

— Boîte à déformation.

— Points d'attraction, de répulsion, effet de vague.

Rendez-vous le mois prochain pour tout savoir sur le format GIF.

Les articles sur les formats d'image, les conversions, et les traitements, ainsi que les sources associés, et toutes les bibliothèques graphiques feront peut-être, prochainement l'objet d'une disquette spéciale que vous pourrez vous procurer par la revue.

Topspeed

Le spécialiste de la compilation

Générateur optimisé Haute qualité du code généré, optimisation de l'édition de liens (toute variable ou fonction non utilisée n'est pas incluse dans l'exécutable).

Plus de limite à 640 Ko Selon que vous choisissez de travailler en mode réel ou mode protégé, la mémoire disponible pour vos programmes peut atteindre de 16 MO à 512 MO. TopSpeed gère automatiquement les overlays pour le code et les données, les librairies dynamiques (DLL) chargées et déchargées à la demande et utilise même le disque en mémoire virtuelle de vos programmes.

Multi-tâches Jusqu'à 32 tâches concurrentes même sous DOS.

Code source disponible Toutes les librairies peuvent être fournies en code source.

Portabilité Génère du code pour DOS, OS/2, et Windows 3.

Multi langages Choisissez le langage le mieux adapté au traitement que vous allez réaliser; TopSpeed permet de construire une application à partir de modules écrits dans des langages differents (C, C++, Pascal, Modula-2). Utiliser n'importe quelle librairie à partir de n'importe quel langage. Développez dans des langages différents en utilisant le même éditeur et le même environnement.

Environnement *TopSpeed* Commun à tous les langages, éditeur 10 fenêtres d'un Mo chacune, gestionnaire de projet, aide hypertexte (langage et librairie), debugger, contrôle de syntaxe, TSR pour calculatrice, table Ascii, scan-code, etc.
DOS ou OS/2 **590,00 F**

TopSpeedC Certifié ANSI pour portabilité maximum, génération automatique des prototypes, compatible Turbo C, Microsoft C 5.1 et Quick C, génère du code compact de haute qualité pour DOS, OS/2 et Windows 3, contrôle des anomalies à l'exécution (pointeur nul, dépassement de bornes, etc).
DOS ou OS/2 **590,00 F**

TopSpeed C++ Conforme AT&T 2.1, programmation orientée objet. Editeur de liens **Smart Method** élimine automatiquement les classes et méthodes non référencées dans le code.
DOS ou OS/2 **590,00 F**

***TopSpeed* Pascal** Conforme ISO 7185 Niveau 1, extensions orientés objets, array et string dynamiques, convertisseur Turbo Pascal, compilation séparée des unités, etc.
DOS ou OS/2 **590,00 F**

***TopSpeed* Modula 2** Successeur de Pascal, langage très structuré et orienté objets, conversion de type automatique.
DOS ou OS/2 **590,00 F**

TopSpeed* TechKit** DLL même sous DOS, Debugger Post-Mortem (permet d'analyser les causes d'un «plantage», Profiler (permet d'optimiser la vitesse d'exécution de votre programme en détectant les sections les plus coûteuses en temps), assembleur ***TopSpeed, compresseur d'EXE, désassembleur, librairies et compilateur de ressources pour Windows 3.
DOS ou OS/2 **590,00 F**

Nouveau ***TopSpeed* DOS Extender**

Fonctionnement en mode protégé sur processeur 80286 et **au delà**. Jusqu'à 512 MO de programmes et données chargés et déchargés automatiquement à la demande de l'application. Supporte XMS, DPMI, VCPI, API OS/2.
DOS ou OS/2 **1180,00 F**

Offre Promotionnel

☐ Mode Réel		☐ Mode Protégé	
Environnement	590,00	Environnement	590,00
+ 1 compilateur	590,00	+ 1 compilateur	590,00
		+ DOS Extender	1180,00
	~~1180,00 HT~~		~~2360,00 HT~~
	990,00 HT		1990,00 HT
+ TVA + frais d'envoi	**1249,14 TTC**		**2435,14 TTC**

Bon de commande

Veuillez trouver ci-joint mon règlement par chèque pour l'offre promotionnelle de ☐ MODE Réel ☐ MODE Protégé – en version anglaise à l'ordre de :

BIONIX – 32, rue Molitor – 75016 Paris
Tél : 40 71 09 36 Fax : 46 51 24 26

TopSpeed® C++ TopSpeed® Modula-2 TopSpeed® Pascal TopSpeed® C

```
02/13/1992  13:38        FICHIER  :  TRAITE.C                    PAGE   1
/*
        TRAITE                                  1.00

        ▪ FONCTION   :  Traitement d'image

        ▪ AUTEUR     :   Jean-Michel SPINER

*/

char *release = "1.00";

#include <stdio.h>
#include <string.h>
#include <dos.h>
#include <malloc.h>

#define MAX_BUFFER 50

#define TRUE 1
#define FALSE 0
#define PCX_EXT ".PCX"
#define LONG_HEADER_PCX 128L
#define INT_VIDEO 0x10
#define TEXT 0x03  /* Mode texte (80x25x16)      */
#define VGA2 0x13  /* VGA Mode 13H (320x200x256) */
#define LONG_PAL_VGA 768L
#define TAILLE_BUFFER 16000
#define MAX_COLOR 256

typedef struct pcxheader {
   unsigned char griffe;
   unsigned char version;
   unsigned char codage;
   unsigned char bits_par_pxl;
   unsigned short x1;
   unsigned short y1;
   unsigned short x2;
   unsigned short y2;
   unsigned short hres;
   unsigned short vres;
   unsigned char pal[48];
   char bidon;
   char nbr_planes;
   unsigned short bytes_par_lgn;
   unsigned short type_pal;
   unsigned short shres;
   unsigned short svres;
   char rabdeplace[54];
};

typedef struct struct_coul {
   unsigned char r,v,b;
};
typedef struct struct_short_coul {
   short r,v,b;
};

typedef struct struct_buffer {
   unsigned short x1;
   unsigned short y1;
   unsigned short w;
```

```
02/13/1992  13:38        FICHIER  :  TRAITE.C                    PAGE   2
   unsigned short h;
   char in_use;
   unsigned char huge *ptr_data;
   struct struct_coul ptr_pal[MAX_COLOR];
};

unsigned short txt_screen[2000];
unsigned short far *ptr_screen;

unsigned short nbr_buffer_en_cours;

struct struct_buffer buffer[MAX_BUFFER];

unsigned char use_bios = FALSE;


/*  ____________________________________________________________  */
/*                          MAIN                                  */
/*  ____________________________________________________________  */
main(argc,argv)
int argc;
char *argv[];
{
   register int i1;
   char fin;
   char cmde[80];

   fin = FALSE;
   FP_SEG (ptr_screen) = 0xB800;
   FP_OFF (ptr_screen) = 0x0000;

   nbr_buffer_en_cours = 0;
   for (i1=0;i1<MAX_BUFFER;i1++) {
      buffer[i1].in_use = FALSE;
   }

   while (fin==FALSE) {
      for (i1=0;i1<MAX_BUFFER;i1++) {
         if (buffer[i1].in_use == TRUE) {
            printf ("\nbuffer %3d   : origine %d,%d   taille %d,%d",
            i1,buffer[i1].x1,buffer[i1].y1,buffer[i1].w,buffer[i1].h);
         }
      }
      printf ("\n>");
      memset (cmde,0,80);
      gets (cmde);
      if (strlen(cmde)!=0) {
         switch (cmde[0]) {
            case 'd' :
            case 'D' :
               Display_buffer (cmde);
               break;
            case 'f' :
            case 'F' :
               Applique_filtre (cmde);
               break;
            case '?' :
            case 'h' :
            case 'H' :
               Display_help ();
               break;
            case 'l' :
            case 'L' :
               Load_buffer (cmde);
```

```
02/13/1992  13:38        FICHIER  :  TRAITE.C                          PAGE   3

                  break;
               case 'm' :
               case 'M' :
                  switch (cmde[1]) {
                     case 'b' :
                     case 'B' :
                        printf ("affichage par bios");
                        use_bios = TRUE;
                        break;
                     case 'd' :
                     case 'D' :
                        printf ("affichage par accès memoire direct");
                        use_bios = FALSE;
                        break;
                     default :
                        printf ("%s : commande inconnue\n",cmde);
                        break;
                  }
                  break;
               case 'q' :
               case 'Q' :
                  fin = TRUE;
                  break;
               case 'r' :
               case 'R' :
                  Resize (cmde);
                  break;
               case 's' :
               case 'S' :
                  Supprime_buffer (cmde);
                  break;
               case 'w' :
               case 'W' :
                  Write_buffer (cmde);
                  break;
               default :
                  printf ("%s : commande inconnue",cmde);
                  break;
            }
         }
      }
      printf ("\n\n");
}



/*                  DISPLAY_HELP                          */
/*  ____________________________________________________  */
Display_help ()
{
   printf ("Liste des commandes disponibles :");
   printf ("\nnum designe le n° du buffer");
   printf ("\nD num : display le buffer num");
   printf ("\nF r|v|b|t|s|l val num : application d'un filtre");
   printf ("\nH ou ? : affichage de l'aide");
   printf ("\nL nom[.pcx] num : chargement du fichier nom ds buffer");
   printf ("\nMB|MD : affichage en mode bios|direct");
   printf ("\nQ : sortie du programme");
   printf ("\nR num1 x y num2 : resize buff1 en dim x/y dans buff2");
   printf ("\nS num : supprime le buffer num (liberation memoire)");
}
```

```
02/13/1992  13:38        FICHIER  :  TRAITE.C                          PAGE   4

/*                  DISPLAY_BUFFER                        */
/*  ____________________________________________________  */
Display_buffer (char *cmde)
{
   short num_buffer;
   unsigned char posx,posy;

   cmde++; cmde++;
   if (strlen(cmde)==0) {
      fprintf (stderr,"Donnez un n° d'buffer");
      return (-1);
   }
   num_buffer = atoi(cmde);
   /* buffer existante ? */
   if (buffer[num_buffer].in_use == FALSE) {
      fprintf (stderr,"buffer %d non initialisee",num_buffer);
      return (-5);
   }
   /* sauve contenu de l'ecran & position */
   memcpy (txt_screen,ptr_screen,4000);
   Get_cur_pos (&posx,&posy);
   set_graph_mode (VGA2);
   Datazone_to_display (num_buffer);
   getch();
   set_graph_mode (TEXT);
   /* restore contenu de l'ecran & position */
   memcpy (ptr_screen,txt_screen,4000);
   Set_cur_pos (posx,posy);
}


/*                  SUPPRIME_BUFFER                       */
/*  ____________________________________________________  */
Supprime_buffer (char *cmde)
{
   short num_buffer;

   cmde++; cmde++;
   if (strlen(cmde)==0) {
      fprintf (stderr,"Donnez un n° d'buffer");
      return (-1);
   }
   num_buffer = atoi(cmde);
   /* buffer existante ? */
   if (buffer[num_buffer].in_use == FALSE) {
      fprintf (stderr,"buffer %d non initialisee",num_buffer);
      return (-2);
   }
   buffer[num_buffer].in_use = FALSE;
   buffer[num_buffer].x1 = 0;
   buffer[num_buffer].y1 = 0;
   buffer[num_buffer].w = 0;
   buffer[num_buffer].h = 0;
   hfree (buffer[num_buffer].ptr_data);
   buffer[num_buffer].ptr_data = NULL;
   memset (buffer[num_buffer].ptr_pal,0,3*MAX_COLOR);
}



/*                  LOAD_BUFFER                           */
/*  ____________________________________________________  */
Load_buffer (char *cmde)
{
```

```
   register int i1;
   char nom[80];
   char *ptr_nom;
   struct pcxheader headpcx;
   short num_buffer;
   FILE *source;
   long l1;

   cmde++; cmde++;
   ptr_nom = nom;
   while (*(cmde)!=' ') *(ptr_nom++) = *(cmde++);
   *ptr_nom = 0;
   num_buffer = atoi(cmde);

   printf ("chargement de %s dans buffer %d",nom,num_buffer);

   /* buffer libre ? */
   if (buffer[num_buffer].in_use == TRUE) {
      fprintf (stderr,"\nbuffer %d non libre",num_buffer);
      return (-5);
   }
   /* Existence fichier */
   if (strstr(strupr(nom),PCX_EXT)==NULL) strcat(nom,PCX_EXT);
   if ((source=fopen(nom,"rb"))==NULL) {
      fprintf (stderr,"\nFichier %s non trouve",nom);
      return (-1);
   }
   fclose (source);

   /* Lecture de l'en-tête */
   if (Read_header_pcx(nom,&headpcx)!=0) {
      fprintf (stderr,"\nErreur lecture header");
      return (-2);
   }
   buffer[num_buffer].x1 = headpcx.x1;
   buffer[num_buffer].y1 = headpcx.y1;
   buffer[num_buffer].w = headpcx.x2-headpcx.x1+1;
   buffer[num_buffer].h = headpcx.y2-headpcx.y1+1;

   /* Alloc memoire */
   l1 = 4096L+((long)buffer[num_buffer].w*(long)buffer[num_buffer].h);
   buffer[num_buffer].ptr_data =(unsigned char huge *) halloc (l1,1);

   if (buffer[num_buffer].ptr_data==NULL) {
      fprintf (stderr,"\nErreur d'allocation memoire");
      return (-3);
   }

   if ((i1=Pcx_to_datazone(nom,buffer[num_buffer].ptr_data,
   buffer[num_buffer].w,buffer[num_buffer].h))!=0) {
      fprintf (stderr,"\nErreur %d dans Pcx_to_datazone",i1);
      return (i1);
   }

   if ((i1=Get_pal_pcx (nom,&buffer[num_buffer].ptr_pal))!=0) {
      fprintf (stderr,"\nErreur %d dans Get_pal_pcx",i1);
      return (i1);
   }

   buffer[num_buffer].in_use = TRUE;
   printf (" ok");
   return (0);
}
```



```
/*                    WRITE_BUFFER                    */
/*  ____________________________________________      */
Write_buffer (char *cmde)
{
   register int i1;
   char nom[80];
   char *ptr_nom;
   struct pcxheader headpcx;
   short num_buffer;
   FILE *cible;
   unsigned char mark;

   cmde++; cmde++;
   ptr_nom = nom;
   while (*(cmde)!=' ') *(ptr_nom++) = *(cmde++);
   *ptr_nom = 0;
   num_buffer = atoi(cmde);
   strcat (nom,PCX_EXT);

   printf ("Sauvegarde dans %s du buffer %d",nom,num_buffer);

   /* buffer libre ? */
   if (buffer[num_buffer].in_use == FALSE) {
      fprintf (stderr,"\nbuffer %d libre",num_buffer);
      return (-5);
   }

   headpcx.griffe = 10;
   headpcx.version = 5;
   headpcx.codage = 1;
   headpcx.bits_par_pxl = 8;
   headpcx.bytes_par_lgn = buffer[num_buffer].w;
   headpcx.x1 = buffer[num_buffer].x1;
   headpcx.y1 = buffer[num_buffer].y1;
   headpcx.x2 = buffer[num_buffer].x1+buffer[num_buffer].w-1;
   headpcx.y2 = buffer[num_buffer].y1+buffer[num_buffer].h-1;
   headpcx.hres = 320;
   headpcx.vres = 200;
   headpcx.nbr_planes = 1;
   headpcx.type_pal = 1;

   if ((cible=fopen(nom,"wb"))==NULL) {
      fprintf (stderr,"\nerreur ouverture %s",nom);
      return (-1);
   }
   if (fwrite(&headpcx,(int)LONG_HEADER_PCX,1,cible)!=1) {
      fprintf (stderr,"\nerreur ecriture %s",nom);
      return (-2);
   }
   fclose (cible);

   Datazone_to_pcx(nom,buffer[num_buffer].ptr_data);

   if ((cible=fopen(nom,"ab"))==NULL) return (1);
   /* palette en fin de fichier */
   fseek (cible,0L,SEEK_END);
   mark = 0x0C;
   fwrite (&mark,1,1,cible);
   fwrite (&buffer[num_buffer].ptr_pal,(int)LONG_PAL_VGA,1,cible);
   fclose (cible);
   printf (" ok");

}
```

```
02/13/1992  13:38        FICHIER  :  TRAITE.C                          PAGE   7

/*              APPLIQUE_FILTRE                                    */
/*  ____________________________________________                   */
Applique_filtre (char *cmde)
{
   register int i1;
   short filtre;
   short val;
   short num_buffer;
   short new_val;
   short decalage;
   char str10[10];
   char *ptr_str;
   unsigned short *ptr_work;
   struct struct_short_coul work1[MAX_COLOR];
   short r,v,b,t,s,l;

   cmde++;
   switch (*cmde) {
      case 'r':
      case 'R':
         filtre = 1;
         break;
      case 'v':
      case 'V':
         filtre = 2;
         break;
      case 'b':
      case 'B':
         filtre = 3;
         break;
      case 't':
      case 'T':
         filtre = 4;
         break;
      case 's':
      case 'S':
         filtre = 5;
         break;
      case 'l':
      case 'L':
         filtre = 6;
         break;
      default :
         fprintf (stderr,"Erreur dans filtre");
         return (-1);
   }
   cmde++;
   cmde++;
   ptr_str = str10;
   while ((*(cmde)!=' ')&&(*(cmde)!=0)) *(ptr_str++) = *(cmde++);
   *ptr_str = 0;
   cmde++;
   val = atoi(str10);

   num_buffer = atoi(cmde);

   /* buffer existante ? */
   if (buffer[num_buffer].in_use == FALSE) {
      fprintf (stderr,"buffer %d non initialisee\n",num_buffer);
      return (-5);
   }

   printf ("filtre de type %d de valeur %d sur buffer %d",filtre,val,num_buffer);
```

```
02/13/1992  13:38        FICHIER  :  TRAITE.C                          PAGE   8

   if (filtre<4) {
      for (i1=0;i1<MAX_COLOR;i1++) {
         work1[i1].r = buffer[num_buffer].ptr_pal[i1].r;
         work1[i1].v = buffer[num_buffer].ptr_pal[i1].v;
         work1[i1].b = buffer[num_buffer].ptr_pal[i1].b;
      }
      decalage = filtre - 1;
      ptr_work = (short *)work1;
      ptr_work += decalage;
      for (i1=0;i1<MAX_COLOR;i1++) {
         new_val = *(ptr_work) + val;
         if (new_val > 255) new_val = 255;
         if (new_val < 0) new_val = 0;
         *(ptr_work) = new_val;
         ptr_work += 3;
      }
      for (i1=0;i1<MAX_COLOR;i1++) {
         buffer[num_buffer].ptr_pal[i1].r = work1[i1].r;
         buffer[num_buffer].ptr_pal[i1].v = work1[i1].v;
         buffer[num_buffer].ptr_pal[i1].b = work1[i1].b;
      }
   }
   else {
      for (i1=0;i1<MAX_COLOR;i1++) {
         Rvb_tsl (buffer[num_buffer].ptr_pal[i1].r,
         buffer[num_buffer].ptr_pal[i1].v,
         buffer[num_buffer].ptr_pal[i1].b,
         &work1[i1].r,&work1[i1].v,&work1[i1].b);
      }
      decalage = filtre - 4;
      ptr_work = (short *)work1;
      ptr_work += decalage;
      for (i1=0;i1<MAX_COLOR;i1++) {
         new_val = *(ptr_work) + val;
         if (new_val > 255) new_val = 255;
         if (new_val < 0) new_val = 0;
         *(ptr_work) = new_val;
         ptr_work += 3;
      }
      for (i1=0;i1<MAX_COLOR;i1++) {
         Tsl_rvb (work1[i1].r,work1[i1].v,work1[i1].b,
         &buffer[num_buffer].ptr_pal[i1].r,
         &buffer[num_buffer].ptr_pal[i1].v,
         &buffer[num_buffer].ptr_pal[i1].b);
      }
   }
   return (0);
}



/*              RVB_TSL                                            */
/*____________________________________________________________*/
Rvb_tsl(int r,int v,int b,int *t,int *s,int *l)
{
   int max,min,d;
   float rc,vc,bc,teinte;

   if (r>v)   max=r; else max=v;
   if (b>max) max=b;
   if (r<v)   min=r; else min=v;
   if (b<min) min=b;
   *l=max;                  /* Luminance */
   d=max-min;               /* Saturation */
   if (max==0) {
```

```
      *s=0;
   }
   else {
      *s=(double)255*d/max;
   }
   if (*s==0) {           /* teinte */
      teinte=0;
   }
   else {
      rc=((double)max-r)/d;
      vc=((double)max-v)/d;
      bc=((double)max-b)/d;
      if (r==max) teinte=bc-vc;
      else if (v==max) teinte=2+rc-bc;
         else teinte=4+vc-rc;
      *t=teinte*42;
      if (*t<0) *t=*t+254;
   }
}




/*              TSL_RVB                          */
/*-----------------------------------------------*/
Tsl_rvb(int t,int s,int l,int *r,int *v,int *b)
{
   int i,p,q,w;
   double T,S,F;

   S=(double)s/256;

   if (S==0) {*r=l;*v=l;*b=l;}
   else {
      T=(double)t/42;
      i=T;
      F=T-i;
      p=l*(1-S);
      q=l*(1-S*F);
      w=l*(1-S*(1-F));
      switch(i){
         case 0 : *r=l; *v=w; *b=p; break;
         case 1 : *r=q; *v=l; *b=p; break;
         case 2 : *r=p; *v=l; *b=w; break;
         case 3 : *r=p; *v=q; *b=l; break;
         case 4 : *r=w; *v=p; *b=l; break;
         case 5 : *r=l; *v=p; *b=q; break;
      }
   }
}




/*              RESIZE                           */
/*-----------------------------------------------*/
Resize (char *cmde)
{
   register int i1;
   register int i2;
   short x_source,y_source;
   short x_cible, y_cible;
   short buffer_source,buffer_cible;
```



```
   char num[80];
   char *ptr_num;
   char *ptr_cible;
   short y;
   int lgn_cible, ay;
   float fy;
   short *buff_source;
   short *buff_cible;
   short **xmap;
   short *ptr;
   unsigned char huge *hptr;
   unsigned short max;
   long l1;

   cmde++; cmde++;
   ptr_num = num;
   while ((*(cmde)!=' ')&&(*(cmde)!=0))
      *(ptr_num++) = *(cmde++);
   *ptr_num = 0;
   buffer_source = atoi(num);
   cmde++;
   ptr_num = num;
   while ((*(cmde)!=' ')&&(*(cmde)!=0))
      *(ptr_num++) = *(cmde++);
   *ptr_num = 0;
   x_cible = atoi(num);
   cmde++;
   ptr_num = num;
   while ((*(cmde)!=' ')&&(*(cmde)!=0))
      *(ptr_num++) = *(cmde++);
   *ptr_num = 0;
   y_cible = atoi(num);
   cmde++;
   ptr_num = num;
   while ((*(cmde)!=' ')&&(*(cmde)!=0))
      *(ptr_num++) = *(cmde++);
   *ptr_num = 0;
   buffer_cible = atoi(num);




   x_source = buffer[buffer_source].w;
   y_source = buffer[buffer_source].h;

   printf ("\nresize %d %d %d %d",buffer_source,x_cible, y_cible,buffer_cible);

   if ((buffer[buffer_source].in_use==FALSE) ||
   (buffer[buffer_cible].in_use==TRUE) ||
   (x_cible<0) || (x_cible>320) ||
   (y_cible<0) || (y_cible>200)) {
      fprintf (stderr,"\nErreur de paramètres\n");
      return (-1);
   }


   buffer[buffer_cible].w = x_cible;
   buffer[buffer_cible].h = y_cible;
   buffer[buffer_cible].x1 = 0;
   buffer[buffer_cible].y1 = 0;
   l1 = 4096L + ((long)x_cible*(long)y_cible);
   buffer[buffer_cible].ptr_data =
```

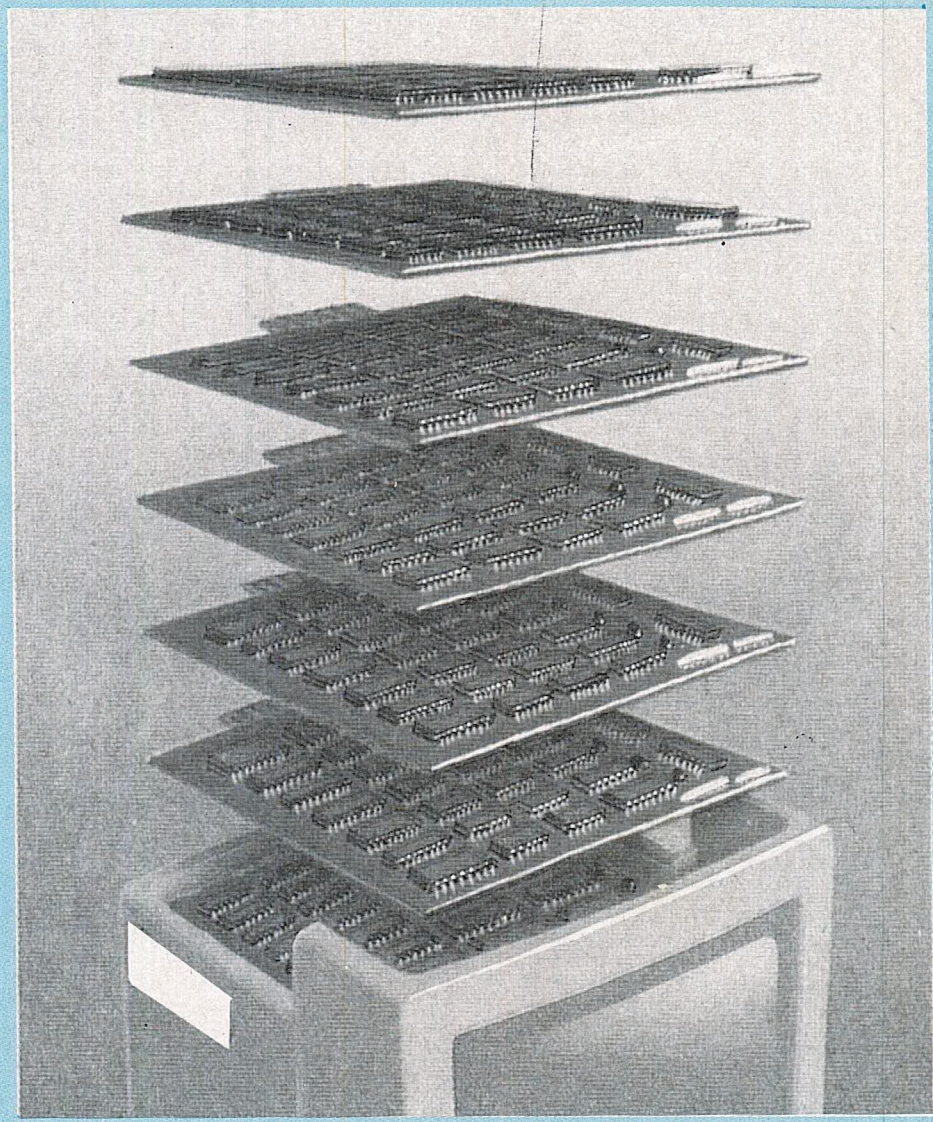
```
02/13/1992  13:38        FICHIER  :  TRAITE.C                    PAGE  11

   (unsigned char huge *) halloc (l1,1);

   if (buffer[buffer_cible].ptr_data==NULL) {
      fprintf (stderr,"\nErreur d'allocation memoire");
      return (-3);
   }
   buffer[buffer_cible].in_use = TRUE;

   memcpy (buffer[buffer_cible].ptr_pal,
   buffer[buffer_source].ptr_pal,MAX_COLOR*3);

   buff_source = (short *)malloc(x_source*sizeof(short));
   buff_cible = (short *)malloc(x_cible*sizeof(short));
   xmap = (short **)malloc(x_cible*sizeof(short *));
   if ((buff_source==NULL)||(buff_cible==NULL)||(xmap==NULL)) {
      fprintf (stderr,"Erreur allocation memoire\n");
      return (-1);
   }
   makexmap(buff_source,xmap,x_source,x_cible);
   lgn_cible = -1;
   for(y=0; y<y_cible; y++) {
      if ((y % 10)==0) printf (".");
      fy = ((float)y+0.5) / (float)y_cible;
      ay = fy * y_source;
      if(lgn_cible != ay) {
         hptr = buffer[buffer_source].ptr_data;
         max = ay*x_source;
         hptr += max;
         ptr = buff_source;
         for (i1=0;i1<x_source;i1++) {
            *(ptr++) = (short)*(hptr++);
         }
         if(x_source != x_cible)
            xscalebuf(xmap,buff_cible,x_cible);
         lgn_cible = ay;
      }
      if(x_source == x_cible) {
         hptr = buffer[buffer_cible].ptr_data;
         max = y*x_cible;
         hptr += max;
         ptr = buff_source;
         for (i1=0;i1<x_cible;i1++)
            *(hptr++) = (unsigned char)*(ptr++);
      }
      else {
         hptr = buffer[buffer_cible].ptr_data;
         max = y*x_cible;
         hptr += max;
         ptr = buff_cible;
         for (i1=0;i1<x_cible;i1++)
            *(hptr++) = (unsigned char)*(ptr++);
      }
   }
   free (buff_source);
   free (buff_cible);
   free (xmap);
}


makexmap(buff_source,xmap,x_source,x_cible)
short *buff_source;
short *xmap[];
int x_source, x_cible;
{
   int x, ax;
   float fx;
```

```
02/13/1992  13:38        FICHIER  :  TRAITE.C                    PAGE  12

   for(x=0; x<x_cible; x++) {
      fx = (x+0.5) / (float)x_cible;
      ax = fx * x_source;
      xmap[x] = buff_source+ax;
   }
}

xscalebuf(xmap,buff_cible,x_cible)
register short *xmap[];
register short *buff_cible;
register int x_cible;
{
   while(x_cible--) {
      if(x_cible >= 7) {
         *buff_cible++ = *(*xmap++);
         *buff_cible++ = *(*xmap++);
         *buff_cible++ = *(*xmap++);
         *buff_cible++ = *(*xmap++);
         *buff_cible++ = *(*xmap++);
         *buff_cible++ = *(*xmap++);
         *buff_cible++ = *(*xmap++);
         *buff_cible++ = *(*xmap++);
         x_cible -= 7;
      }
      else {
         *buff_cible++ = *(*xmap++);
      }
   }
}



/*          DATAZONE_TO_DISPLAY                          */
/*  ____________________________________________________ */
Datazone_to_display(num_buffer)
{
   int i1;
   register unsigned short col;
   register unsigned short lgn;
   unsigned char huge *ptr_video;


   for (i1=0;i1<MAX_COLOR;i1++) {
      Set_palette_color (i1,buffer[num_buffer].ptr_pal[i1].r,
      buffer[num_buffer].ptr_pal[i1].v,buffer[num_buffer].ptr_pal[i1].b);
   }
   ptr_video = buffer[num_buffer].ptr_data;

   for (lgn=0;lgn<buffer[num_buffer].h;lgn++)
      for (col=0;col<buffer[num_buffer].w;col++)
         if (use_bios==TRUE)
            Put_pixel_bios (col,lgn,*(ptr_video++));
         else
            Put_pixel_32 (col,lgn,*(ptr_video++));
}



/*                PRIMITIVES GRAPHIQUES                      */
/*  ____________________________________________________ */
```

```
Get_graph_mode()
{
   union REGS inregs,outregs;

   inregs.h.ah = 0x0F;
   int86 (INT_VIDEO,&inregs,&outregs);
   outregs.h.ah = 0x00;
   return (outregs.x.ax);
}



Set_graph_mode (graph_mode)
unsigned short graph_mode;
{
   union REGS inregs,outregs;

   inregs.h.ah = 0;
   inregs.h.al = (unsigned char)graph_mode;
   int86 (INT_VIDEO,&inregs,&outregs);
   return (0);
}

Get_cur_pos (x,y)
unsigned char *x,*y;
{
   union REGS inregs,outregs;

   inregs.h.ah = 0x03;
   inregs.h.bh = 0x00;
   int86 (INT_VIDEO,&inregs,&outregs);
   *x = outregs.h.dl;
   *y = outregs.h.dh;
}

Set_cur_pos (x,y)
unsigned char x,y;
{
   union REGS inregs,outregs;

   inregs.h.ah = 0x02;
   inregs.h.bh = 0x00;
   inregs.h.dl = x;
   inregs.h.dh = y;
   int86 (INT_VIDEO,&inregs,&outregs);
}

Set_Palette_Color (num,r,v,b)
unsigned char num,r,v,b;
{
   union REGS inregs,outregs;

   inregs.x.bx = num;
   inregs.h.dh = r>>2;
   inregs.h.ch = v>>2;
   inregs.h.cl = b>>2;
   inregs.h.ah = 0x10;
   inregs.h.al = 0x10;
   int86 (INT_VIDEO,&inregs,&outregs);
   return (0);
}


Put_pixel_bios (x,y,couleur)
unsigned short x,y,couleur;
{
```



```
   union REGS inregs,outregs;

   inregs.h.ah = 0x0C;
   inregs.h.al = (unsigned char)couleur;
   inregs.h.bh = 0x00;
   inregs.x.cx = x;
   inregs.x.dx = y;
   int86 (INT_VIDEO,&inregs,&outregs);
   return (0);
}
```



```
02/13/1992  13:22        FICHIER  :  PCX13H.C                          PAGE   1

/*
        |                    PCX13H                    |  1.00  |
        |                                                         |
        |    ▪ FONCTION    :  Lib pour PCX en mode                |
        |                      13h (320x200 en 256 couleurs)      |
        |                                                         |
        |    ▪ AUTEUR      :  Jean-Michel SPINER                  |
        |                                                         |
*/

#include <dos.h>
#include <stdio.h>
#include <malloc.h>

#define TRUE 1
#define FALSE 0
#define LONG_HEADER_PCX 128L
#define LONG_PAL_VGA 768L
#define PCX_EXT ".PCX"
#define INT_VIDEO 0x10
#define TAILLE_BUFFER 16000


typedef struct pcxheader {
   unsigned char griffe;
   unsigned char version;
   unsigned char codage;
   unsigned char bits_par_pxl;
   unsigned short x1;
   unsigned short y1;
   unsigned short x2;
   unsigned short y2;
   unsigned short hres;
   unsigned short vres;
   unsigned char pal[48];
   char bidon;
   char nbr_planes;
   unsigned short bytes_par_lgn;
   unsigned short type_pal;
   unsigned short shres;
   unsigned short svres;
   char rabdeplace[54];
};


typedef struct struct_coul {
   unsigned char r,v,b;
};


unsigned char buffer_lecture[TAILLE_BUFFER];



/*                          PROTOTYPES                          */

int Read_header_pcx (char *,struct pcxheader *);
int Pcx_to_datazone(char *,unsigned char huge *,unsigned short,unsigned short);
int Get_pal_pcx (char *,unsigned char *);
int Write_header_pcx (char *,struct pcxheader *);
int Datazone_to_pcx (char *,unsigned char huge *);
```

```
02/13/1992  13:22        FICHIER  :  PCX13H.C                          PAGE   2



/*                  READ_HEADER_PCX                          */
/*  ____________________________________________________    */
Read_header_pcx (char *infile,struct pcxheader *headpcx)
{
   FILE  *source;

   if ((source=fopen(infile,"rb"))==NULL) return (1);
   if (fread(headpcx,(int)LONG_HEADER_PCX,1,source)!=1) return (2);
   fclose (source);
   return (0);
}



/*              PCX_TO_DATAZONE                              */
/*  ____________________________________________________    */
Pcx_to_datazone(infile,ptr_video,w,h)
char *infile;
unsigned char huge *ptr_video;
unsigned short w,h;
{
   register unsigned short i1;
   register unsigned short i2;
   unsigned short ptr;
   unsigned char val;
   unsigned short nbr_bytes;
   unsigned short x,y;
   unsigned short nb_carlu;
   unsigned char dernier_bloc;
   FILE  *source;

   x=0; y=0;
   if ((source=fopen(infile,"rb"))==NULL) return (1);
   fseek (source,LONG_HEADER_PCX,SEEK_SET);
   dernier_bloc = FALSE;
   while (dernier_bloc == FALSE) {
      if ((nb_carlu=fread(buffer_lecture,1,TAILLE_BUFFER,source))!=TAILLE_BUFFER)

         dernier_bloc=TRUE;
      ptr = 0;
      while (ptr<nb_carlu) {
         if ((buffer_lecture[ptr] & 0xC0) == 0xC0) {
            nbr_bytes = buffer_lecture[ptr] & 0x3F;
            ptr++;
         }
         else {
            nbr_bytes = 1;
         }
         val = buffer_lecture[ptr];
         if (y<=h) {
            for (i1=0;i1<nbr_bytes;i1++) {
               *(ptr_video++) = val;
               x++;
               if (x>=w) {
                  x = 0;
                  y++;
               }
            }
         }
         ptr++;
```

```
02/13/1992  13:22        FICHIER  :  PCX13H.C                    PAGE   3

      }
   }
   fclose (source);
   return (0);
}



/*                    GET_PAL_PCX                          */
/*   ____________________________________________________  */
Get_pal_pcx (char *infile,unsigned char *ptr_pal)
{
   register int i1;
   FILE *source;

   if ((source=fopen(infile,"rb"))==NULL) return (1);
   /* palette en fin de fichier */
   fseek (source,-LONG_PAL_VGA,SEEK_END);
   fread (ptr_pal,(int)LONG_PAL_VGA,1,source);

   fclose (source);
   return (0);
}


/*                    PUT_PAL_PCX                          */
/*   ____________________________________________________  */
Put_pal_pcx (char *infile,unsigned char *ptr_pal)
{
   register int i1;
   FILE *cible;

   if ((cible=fopen(infile,"ab"))==NULL) return (1);
   /* palette en fin de fichier */
   fseek (cible,0L,SEEK_END);
   fwrite (ptr_pal,(int)LONG_PAL_VGA,1,cible);
   fclose (cible);
   return (0);
}



/*              DATAZONE_TO_PCX                          */
/*   ____________________________________________________  */
Datazone_to_pcx(outfile,ptr_video)
char *outfile;
unsigned char huge *ptr_video;
{
   register unsigned short i1;
   register unsigned short i2;
   unsigned short ptr;
   unsigned char cur_val;
   unsigned char old_val;
   unsigned char compt;
   unsigned short nbr_bytes;
   unsigned short nb_carlu;
   unsigned short w,h;
   unsigned char dernier_bloc;
   FILE  *cible;
   struct pcxheader headpcx;

   if (Read_header_pcx(outfile,&headpcx)!=0) return (1);
   w = headpcx.x2 - headpcx.x1;
```

```
02/13/1992  13:22        FICHIER  :  PCX13H.C                    PAGE   4

   h = headpcx.y2 - headpcx.y1;

   if ((cible=fopen(outfile,"ab"))==NULL) return (1);
   fseek (cible,LONG_HEADER_PCX,SEEK_SET);

   ptr = 0;
   for (i1=0;i1<=h;i1++) {
      for (i2=0;i2<=w;i2++) {
         cur_val = *(ptr_video++);
         if (i2==0) {
            compt = 1;
            old_val = cur_val;
         }
         else {
            if (cur_val!=old_val) {
               if (compt==1) {
                  if (old_val>=0xC0) {
                     buffer_lecture[ptr++]=0xC1;
                     buffer_lecture[ptr++]=old_val;
                  }
                  else {
                     buffer_lecture[ptr++]=old_val;
                  }
               }
               else {
                  buffer_lecture[ptr++]=0xC0+compt;
                  buffer_lecture[ptr++]=old_val;
               }
               old_val = cur_val;
               compt = 1;
            }
            else {
               compt++;
               if (compt>=63) {
                  buffer_lecture[ptr++]=0xC0+compt-1;
                  buffer_lecture[ptr++]=cur_val;
                  compt = 1;
               }
            }
         }
      }

      if (compt==1) {
         if (cur_val>=0xC0) {
            buffer_lecture[ptr++]=0xC1;
            buffer_lecture[ptr++]=cur_val;
         }
         else {
            buffer_lecture[ptr++]=cur_val;
         }
      }
      else {
         buffer_lecture[ptr++]=0xC0+compt;
         buffer_lecture[ptr++]=cur_val;
      }

      if (ptr!=0)
         fwrite (buffer_lecture,ptr,1,cible);
      ptr = 0;
   }

   fclose (cible);
   return (0);
}
```

```
02/13/1992  13:38       FICHIER  :  PUTPIXEL.ASM                    PAGE   1

;;;
```

```
02/13/1992  13:38        FICHIER  :  PUTPIXEL.ASM                    PAGE   3

   _Put_Pixel_62          proc      Near
   sframe       struc
                dw        ?
                dw        ?
ENDIF
x62          dw        ?
y62          dw        ?
val62        dw        ?
sframe       ends
frame        equ       [bp]
             push      bp
             mov       bp,sp

             push      ax
             push      bx
             push      dx
             push      di
             push      ds

             mov       di,frame.y62
             and       di,0FFFEh
             shl       di,1
             shl       di,1
             shl       di,1
             mov       ax,di
             shl       di,1
             shl       di,1
             add       di,ax

             ; ligne paire ?
             test      frame.y62,1
             jz        p62paire
             add       di,2000h
p62paire:
             mov       ax,frame.x62
             mov       bx,ax
             shr       ax,1
             shr       ax,1
             shr       ax,1
             add       di,ax

             and       bx,07h
             mov       bh,80h
p62bra01:
             cmp       bl,00h
             je        p62bra02
             shr       bh,1
             dec       bl
             jmp       p62bra01
p62bra02:
             mov       ax,0B800h
             mov       ds,ax

             ;  MODE MOV
             mov       al,byte ptr [di]
             test      frame.val62,1
             jz        p62noir
             or        al,bh
             jmp       p62ok
p62noir:
             not       bh
             and       al,bh
p62ok:
             mov       byte ptr [di],al

             pop       ds
```

```
02/13/1992  13:38        FICHIER  :  PUTPIXEL.ASM                    PAGE   4

             pop       di
             pop       dx
             pop       bx
             pop       ax

             mov       sp, bp
             pop       bp
             ret

_Put_Pixel_62          endp



;                    PUT_PIXEL_63
;  ____________________________________________________________
IFDEF        LARGE
   _Put_Pixel_63          proc      Far
   sframe       struc
                dw        ?
                dd        ?
ELSE
   _Put_Pixel_63          proc      Near
   sframe       struc
                dw        ?
                dw        ?
ENDIF

x63          dw        ?
y63          dw        ?
val63        dw        ?
sframe       ends
frame        equ       [bp]
             push      bp
             mov       bp,sp

             ; Empilages
             push      di
             push      si
             push      ds

             ; Calcul adresse  --->  di
             mov       ax,frame.y63
             shl       ax,4
             mov       di,ax
             shl       ax,2
             add       di,ax
             mov       ax,frame.x63
             mov       cx,ax
             shr       ax,3
             add       di,ax

             ; Constitution du mask   ---> ch
             and       cl,07h
             mov       ch,80h
             ror       ch,cl

             ; faire pointer ds sur segment video
             push      ds
             mov       ax,0A000h
             mov       ds,ax

             ; Initialiser le mask
             mov       dx,03CEh
             mov       al,08h
```

```
02/13/1992  13:38        FICHIER  :  PUTPIXEL.ASM                    PAGE   5

            out       dx,al
            inc       dx
            mov       al,ch
            out       dx,al

            ; Autorizer ts les plans
            mov       dx,03C4h
            mov       al,02h
            out       dx,al
            inc       dx
            mov       al,0Fh
            out       dx,al

            ; lire
            mov       bl,byte ptr[di]
            ; ecrire
            mov       byte ptr [di],00h

            ; Autorizer les bons plans
            mov       dx,03C4h
            mov       al,02h
            out       dx,al
            inc       dx
            pop       ds
            mov       ax,frame.val63
            out       dx,al

            ; ecrire
            mov       ax,0A000h
            mov       ds,ax
            mov       byte ptr [di],0FFh

            ; REMETTRE LE MASK
            mov       dx,03CEh
            mov       al,08h
            out       dx,al
            inc       dx
            mov       al,0FFh
            out       dx,al


            pop       ds
            pop       si
            pop       di

            mov       sp, bp
            pop       bp
            ret

_Put_Pixel_63         endp




;                    PUT_PIXEL_86
; ____________________________________________________________
IFDEF       LARGE
   _Put_Pixel_86         proc      Far
   sframe       struc
                dw        ?
                dd        ?
```

```
02/13/1992  13:38        FICHIER  :  PUTPIXEL.ASM                    PAGE   6

ELSE
   _Put_Pixel_86         proc      Near
   sframe       struc
                dw        ?
                dw        ?
ENDIF

x86         dw        ?
y86         dw        ?
val86       dw        ?
sframe      ends
frame       equ       [bp]
            push      bp
            mov       bp,sp

            ; Empilages
            push      di
            push      si
            push      ds

            ; Calcul adresse  --->  di
            mov       ax,frame.y86
            shl       ax,2
            mov       di,ax
            shl       ax,3
            add       di,ax
            shl       ax,1
            add       di,ax
            mov       ax,frame.x86
            mov       cx,ax
            shr       ax,3
            add       di,ax

            ; Constitution du mask   ---> ch
            and       cl,07h
            mov       ch,80h
            ror       ch,cl

            ; faire pointer ds sur segment video
            push      ds
            mov       ax,0A000h
            mov       ds,ax

            ; Initialiser le mask
            mov       dx,03CEh
            mov       al,08h
            out       dx,al
            inc       dx
            mov       al,ch
            out       dx,al

            ; Autorizer ts les plans
            mov       dx,03C4h
            mov       al,02h
            out       dx,al
            inc       dx
            mov       al,0Fh
            out       dx,al

            ; lire
            mov       bl,byte ptr[di]
            ; ecrire
            mov       byte ptr [di],00h

            ; Autorizer les bons plans
            mov       dx,03C4h
```

```
            mov     al,02h
            out     dx,al
            inc     dx
            pop     ds
            mov     ax,frame.val86
            out     dx,al

            ; ecrire
            mov     ax,0A000h
            mov     ds,ax
            mov     byte ptr [di],0FFh

            ; REMETTRE LE MASK
            mov     dx,03CEh
            mov     al,08h
            out     dx,al
            inc     dx
            mov     al,0FFh
            out     dx,al


            pop     ds
            pop     si
            pop     di

            mov     sp, bp
            pop     bp
            ret

_Put_Pixel_86       endp




IFDEF   LARGE
    PUTPIXEL_text       ends
ELSE
    text        ends
ENDIF

end
```

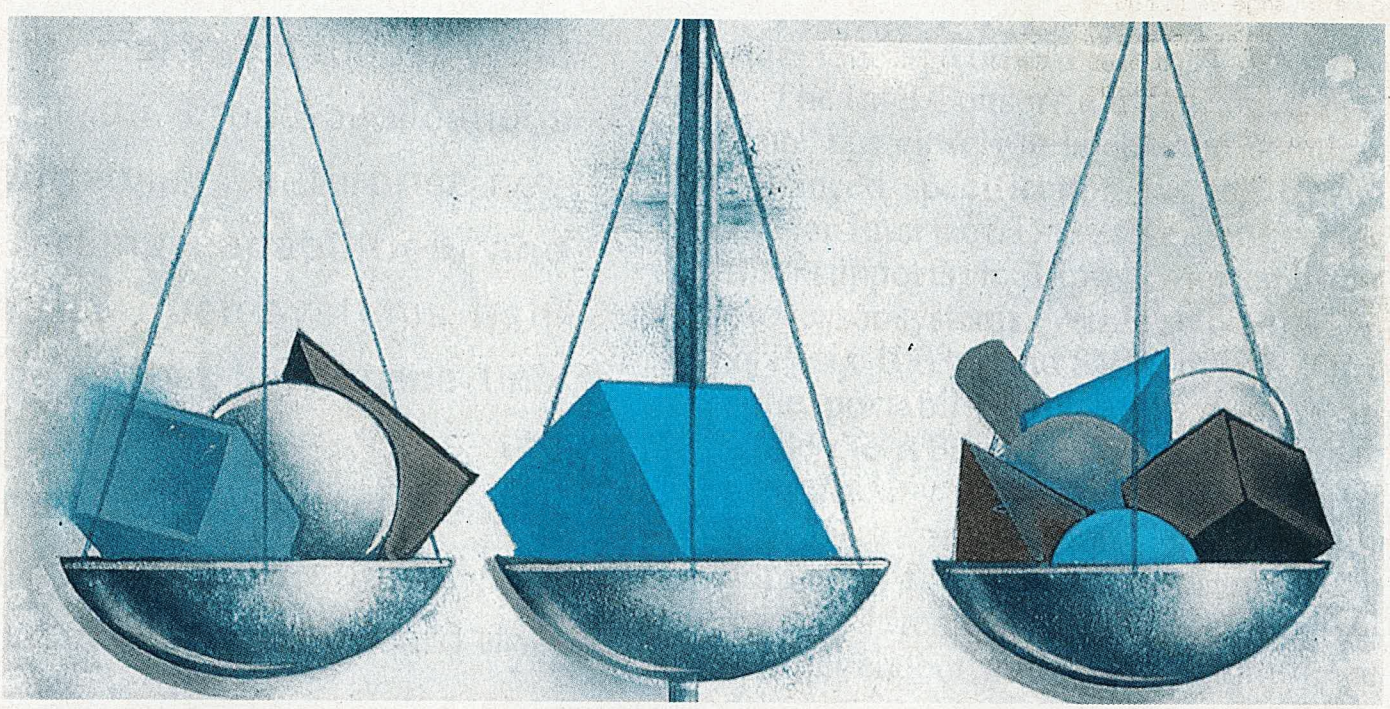


```
#
#               MAKE traite
#

.c.obj:
        cl -c -W1 -Ze -AL -G2 $*.c

.asm.obj:
       masm /DLARGE $*.asm,$*.obj,nul.lst,nul.crf;

putpixel.obj : putpixel.asm

traite.obj : traite.c

pcx13h.obj : pcx13h.c

traite.exe : traite traite.obj pcx13h.obj putpixel.obj
link /ST:5000 traite+pcx13h+putpixel,,,;
```

Clipper 5.01 : la gestion de la mémoire

Les techniques de gestion de la mémoire pendant l'exécution ont beaucoup changé depuis Clipper 87. Le VMM (Gestionnaire de Mémoire Virtuelle) a fait son apparition. De nombreuses options de compilation et d'édition de liens ont bien entendu suivi. Voici un tour d'horizon, non exhaustif, de ces nouvelles fonctionnalités.

Dominique BRUYERE



La plupart des renseignements que je vous livre, sont dans la documentation de Clipper, sous la forme du Norton Guide. Comme je sais que vous préférez une information compactée, j'ai donc repris en partie cette information. Je vous engage néanmoins à utiliser le Norton Guide et à passer du temps à étudier les exemples livrés avec Clipper 5 et qui devraient se trouver dans votre répertoire\CLIPPER5\SOURCE\SAMPLE.

Le Virtual Memory Manager (VMM) ou Gestionnaire de Mémoire Virtuelle

Le terme ''mémoire virtuelle'' est un terme générique pour les techniques matérielles ou logicielles qui permettent à partir d'une quantité limitée de mémoire ''réelle'' d'émuler une plus grande quantité de mémoire virtuelle. Dans cet article, nous y ferons référence par l'abréviation VM.

Le VMM de Clipper 5.01 est implémenté dans le logiciel. Le système tourne en mode réel avec n'importe quel processeur compatible 8086 ; il n'utilise pas directement le mode protégé de mémoire virtuelle des processeurs Intel 286 et 386.

Le VMM est gestionnaire de mémoire ''segmentée''. La mémoire virtuelle est allouée par segments, chacun de ceux-ci pouvant contenir de 1 K à 64 K de données. Initialement, le VMM travaille à peu près comme un gestionnaire de mémoire normal (bien qu'il ait sur la plupart un avantage supplémentaire la place occupée par la plus grande partie des segments est modifiable ; le VMM peut les réorganiser pour optimiser la mémoire réelle).

A l'opposé d'un gestionnaire de mémoire normal, le VMM ne ''plante pas'' quand la mémoire réelle disponible ne peut plus contenir tous les segments alloués. En fait, un ou plusieurs des segments les moins utilisés depuis un certain temps sont ''swappés'' dans un lieu de stockage secondaire, ou en d'autres termes enlevés de la mémoire réelle, pour laisser la place à de nouveaux segments. Plus tard, si l'on a besoin d'accéder à des segments qui ont été ''swappés'', ils sont rappelés et remplacent des segments qui ne sont plus utiles à ce moment précis.

L'espace maximum adressable par le VMM de Clipper 5.01 est de 64 MO !

Utilisation de l'EMM à la norme LIM 3.2 (mémoire étendue)

Par défaut, le gestionnaire de mémoire

de Clipper 5.01 utilise la mémoire étendue à la norme LIM 3.2 (Lotus, Intel, Microsoft) comme zone de stockage secondaire. L'EMM permet un accès très rapide aux segments "swappés". Un maximum de 8 MO d'EMM peut être utilisé.

Configuration : l'utilisation de l'EMM peut être contrôlée à l'aide du paramètre E dans les variables d'environnement de Clipper. Les quantités dont peut disposer l'EMM vont de zéro à 8 192 par unités de 1 K. Un paramètre zéro empêche l'utilisation de l'EMM. Désactiver l'EMM n'est pas recommandé sous Clipper 5.01.

Bien que l'EMM ne soit pas indispensable, elle améliore les performances de la plupart des applications. Dans quelque cas, il peut être bénéfique de limiter l'usage de l'EMM.

Ces cas sont exposés ci-dessous. En effet, plus il y a de mémoire étendue disponible, plus il faut de place en mémoire réelle pour maintenir les tables du système !

Exemple : SET CLIPPER = E2048

Exemple : SET CLIPPER = BADCACHE

Configuration : quelques programmes de cache-disque utilisent l'EMM comme cache de secteurs du disque. Certains programmes de cache-disque peuvent causer des conflits avec le VMM de Clipper 5 s'ils assument que l'application en cours n'utilise pas l'EMM. Le paramètre BADCACHE dans l'environnement de Clipper permet au VMM de sauvegarder et de restaurer l'état de la page frame avant et après chaque accès à l'EMM.

La page-frame de l'EMM est une zone mémoire avec adressage en mode réel à partir de laquelle les données contenues dans l'EMM sont adressées.

Ceci pourrait conduire à la non détection par d'autres programmes utilisant l'EMM de la présence de Clipper dans cette zone. (Note : dans quelques systèmes d'EMM le paramètre BADCACHE peut dégrader les performances du VMM. Il ne devrait être utilisé que si vous avez à faire face à des corruptions de fichiers ou de disques à cause d'un conflit avec un cache-disque ou un autre programme résident).

Le swapping sur disque

Si la quantité d'EMM disponible est insuffisante pour contenir les segments swappés, un fichier disque temporaire (le swap file) est créé pour les recevoir. Par défaut, le VMM utilise un maximum de 8 MO d'espaces disque pour le fichier de swap. L'espace disque n'est utilisé que quand cela s'avère nécessaire.

Configuration : la taille maximale du fichier de swap peut être contrôlée en utilisant le paramètre SWAPK parmi

les variables d'environnement de Clipper. La valeur de ce paramètre va de 256 à 65 535 (256 KO à 64 MO !). Le paramètre SWAPK n'a pas d'effet sur l'activité de swapping du VMM ; il sert, en fait, à établir une limite haute pour la taille du fichier de swapping. Si la limite est dépassée, le VMM arrêtera l'application. La valeur par défaut de ce paramètre est de 16 MO.

Exemple : SET CLIPPER = SWAPK : 26000

Configuration : le disque et le répertoire où le fichier de swap est créé peuvent être contrôlés à l'aide du paramètre SWAPPATH. Par défaut, le fichier de swapping est créé dans le répertoire courant du DOS.

Exemple : SET CLIPPER = SWAPPATH : "C:\SWAP"

Espace de swap

Le terme d'"espace de swap" fait référence à la région de mémoire réelle où le VMM charge les segments de mémoire virtuelle.

Configuration : le VMM occupe l'espace de swap au démarrage de l'application. Par défaut, toute la mémoire DOS disponible est utilisée pour l'espace de swap. L'utilisation de la mémoire réelle peut être réduite par l'utilisation du paramètre X.

Ce paramètre signifie le nombre de KO à exclure de la zone accessible par le VMM. Les paramètres pour cette limitation de mémoire vont de 0 à 256. Le paramètre X est fourni pour conserver une partie de la mémoire DOS pour d'autres usages. (Par exemple des programmes résidents). En général, son utilisation n'est pas conseillée, car il limite l'espace de swap et réduit les performances du VMM.

Exemple : SET CLIPPER = X8

Objet mémoire

Un objet mémoire est un type spécial de gestionnaire de mémoire destiné à gérer des valeurs de données complexes comme les chaînes caractères et les tableaux.

L'objet mémoire de Clipper 5 est appelé le Segmented Virtual Objet Store, (SVOS) et en français me direz-vous, Stockage Virtuel Segmenté. Le SVOS utilise la mémoire virtuelle pour stocker les données, y compris les chaînes caractères, les tableaux, et les codes-blocks créés dynamiquement (compilés à l'exécution).

Le SVOS apporte deux importantes fonctions supplémentaires aux possibilités offertes par le VMM.

— Compactage de la mémoire : les données stockées sont automatiquement compactées selon une méthode continue. Ceci élimine la fragmentation de la mémoire virtuelle et réduit les besoins de swap.

— Collecte d'objets computérisés non identifiés (CONI) : certaines données de Clipper 5 (par exemple les tableaux), peuvent être adressées par plusieurs variables programme (ou éléments de tableau) en même temps. Les algorithmes de récupération des données égarées récupèrent automatiquement l'espace occupé par les valeurs "inaccessibles", c'est-à-dire les

valeurs qui ne sont plus accessibles à partir d'une variable ou d'un tableau.

Pour réduire les retards créés par le compactage de la mémoire et la récupération des valeurs inaccessibles, les algorithmes du SVOS sont ajustés dynamiquement sur la base de leur type, leur taille, sur leur durée de vie en tant que données stockées, la quantité de mémoire disponible (réelle et EMM), et les performances générales du système.

Les algorithmes essayent de gérer la mémoire de façon que le VMM puisse écarter les données peu utilisées, et libérer un espace de mémoire réellement adressable pour les opérations urgentes.

La nature dynamique des algorithmes du SVOS rend difficile un calcul exact de ses capacités maximum. Dans le pire des cas, elles dépassent le MO. Théoriquement, sa capacité dépasse les 16 MO.

Le recouvrement de lien dynamique

Rtlink, le linker fourni avec Clipper 5, ou d'autres linkers du marché comme Blinker ou Warplink, gèrent le code compilé par Clipper en utilisant une technique appelée : overlay dynamique. Cette technique supprime le casse-tête que constituait la création de structures d'overlays compliquées et permet une gestion souple du code compilé pendant l'exécution.

Durant l'édition de liens, Rtlink casse les modules compilés en "pages" d'une longueur fixe. Ces pages sont stockées soit dans le fichier exécutable, soit dans un fichier d'overlay externe. L'architecture par pages élimine les restrictions de mémoire et les calculs basés sur la taille des fonctions ou modules compilés. Les modules importants sont divisés en pages multiples ; les petites fonctions sont, elles, regroupées dans une seule page.

Au moment de l'exécution, le gestionnaire d'overlay dynamique charge des pages en suivant l'information incluse dans le .EXE par l'éditeur de liens. Les pages dynamiques sont chargées dans des segments de mémoire virtuelle, permettant au VMM de gérer les pages d'overlay de façon compétitive avec les autres utilisations de la mémoire.

L'architecture paginée permet au système d'écarter les portions de code les moins utilisées, même si ce code est associé à une fonction active ; il n'est plus nécessaire d'utiliser des overlays imbriqués. Les pages de code qui sont utilisées intensivement sont conservées en mémoire par le VMM suivant la technique du LRU (Last Recently Used, Dernier Utilisé Récemment).

Toutes les fois que cela est possible, le VMM place les pages dynamiques dans l'EMM et réduit de ce fait les temps de lecture. Les pages d'overlay ne sont jamais écrites dans le fichier de swap sur disque. Si un segment contenant une page d'overlay doit être enlevé de la mémoire virtuelle, il est simplement écarté. Si son usage redevient nécessaire, il est lu à nouveau dans le fichier d'overlay.

Configuration : le gestionnaire d'overlay dynamique ne maintiendra les pages d'overlay en mémoire virtuelle que tant qu'il pourra conserver un handle actif pour le fichier qui contient ces pages. Si le fichier est fermé, les pages sont écartées de la mémoire virtuelle. Le nombre maximum de handles utilisés par le gestionnaire d'overlay dynamique peut être contrôlé à l'aide du paramètre DYNF. Les valeurs possibles vont de 1 à 8. La valeur par défaut est de 2. Pour des applications qui utilisent plusieurs fichiers d'overlays externes, le fait d'augmenter cette valeur peut améliorer les performances.

Exemple : SET CLIPPER = DYNF:4

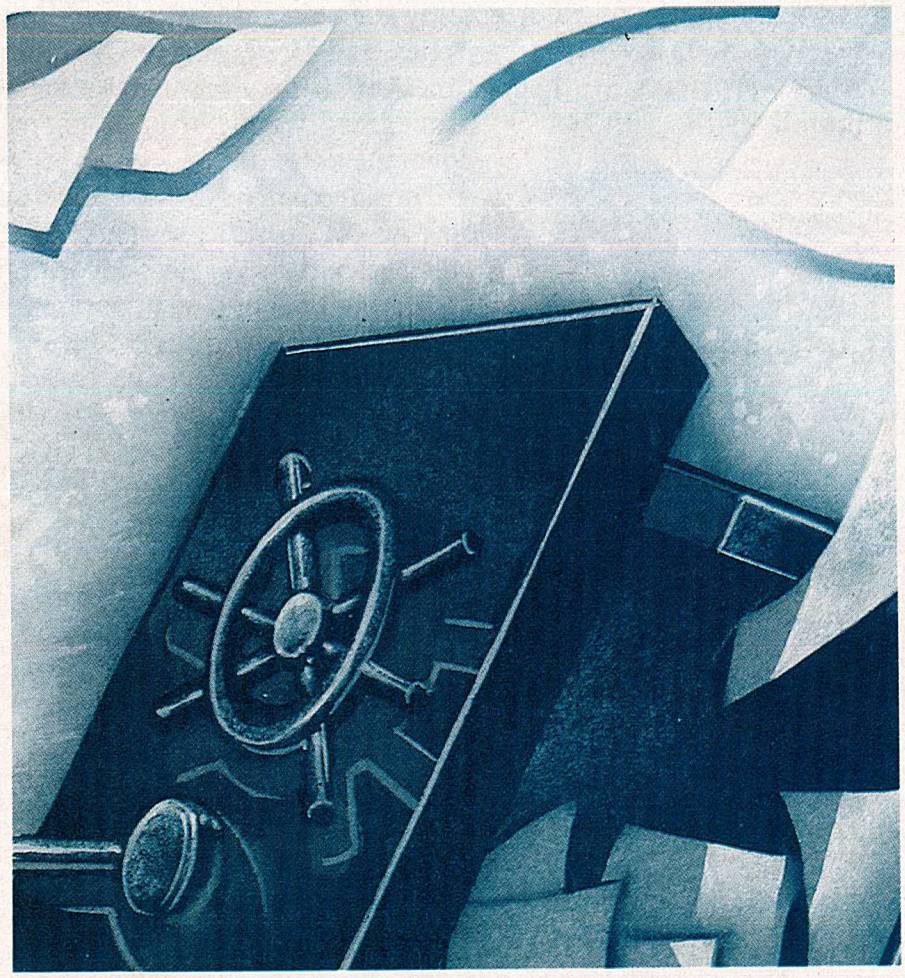
La bufferisation des données fichiers

Les drivers contenus dans DBFNTX.LIB (en attendant la venue de drivers pour Paradox ou autre) utilisent des segments de mémoire virtuelle pour contenir les buffers de fichiers de données et de fichiers d'index.

Ceci permet au VMM de gérer ces buffers sans conflit avec les autres utilisations de la mémoire. Tout comme avec les segments d'overlays dynamiques le VMM swappe les segments de buffer dans l'EMM mais jamais dans le fichier de swap sur disque. Si un segment de VM contenant un buffer de disque doit être retiré de la mémoire, il est réécrit dans le fichier de données approprié. Cette écriture peut être forcée par la commande COMMIT ou la fonction DBCommit(), avec l'avantage, en réseau, de rendre immédiate la mise à jour des fichiers sur disque, et par là même de rendre visibles aux autres processus les modifications apportées le cas échéant.

Le système de bufferisation de fichier alloue des tables de gestion de buffer dans la mémoire fixe (voir le Gestionnaire de Mémoire Fixe ci-dessous). Le total utilisé dépend du total combiné de mémoire fixe et d'EMM utilisable au lancement de l'application. Les tables prennent 14 octets pour chaque 1 K de mémoire réelle ou étendue (exTended).

Les tables ne dépassent jamais 8 K, néanmoins. Parce que ces tables sont situées dans la mémoire fixe, elles ont



un effet direct sur la quantité d'espace disponible pour la zone, de swap disponible pour le VMM. Dans quelques situations critiques d'espace mémoire très restreint. La réduction de l'espace adressable par le VMM, peut vous éviter le message désagréable : ''Swap space exhausted''. Mais encore une fois, et il faut jouer serré avec Clipper 5.01, cette réduction d'espace risque de ralentir considérablement la vitesse d'exécution.

Fermeture de segments

De temps en temps, certains segments de VM sont ''fermés'' par un sous-système. Cela empêche le VMM de bouger ou de swapper ces segments jusqu'à ce que le sous-système les débloque. Cette possibilité est utilisée rarement pour éviter à l'espace de swap de se retrouver encombré par des segments fermés.

Gestionnaire de mémoire fixe

Les sous-systèmes de Clipper 5.01 utilisent la mémoire virtuelle pour stocker le code ou les données qui ont besoin d'être gérés dynamiquement. Certaines allocations ne sont pas appropriées à la VM (par exemple les tables qui doivent rester à la même place durant toute la durée de l'application). Un gestionnaire de mémoire fixe dans le style de ceux rencontrés en C est inclus dans Clipper 5.01.

Le système de mémoire fixe attribut un ou plusieurs segments de VM et les bloque d'une façon permanente dans la mémoire fixe. L'espace entre ces segments est ensuite partagé en façon des besoins. Puisque ces segments fixes ne sont jamais déverrouillés, le VMM les place tout en bas de l'espace de swap pour empêcher qu'ils n'interfèrent avec d'autres opérations.

Les tables de variable mémoire

Clipper 5.01 apporte différentes solutions de stockage des variables programme. Les variables LOCAL et STATIC sont stockées dans un espace réservé de la mémoire réelle. Les variables mémoire PUBLIC et PRIVATE, sont stockées dans des segments de VM.

Comme les variables mémoire sont créées et détruites dynamiquement pendant l'exécution du programme, les segments associés grossissent dynamiquement pendant l'exécution, dans un rapport direct avec le nombre de ces variables en service.

Pour des raisons de performance, ces segments restent bloqués pendant la plupart des opérations (ils sont débloqués pendant les commandes RUN, (à éviter) et certaines commandes dévoreuses de mémoire). Dans les situations de quantité de mémoire précaire, les tables de variables mémoire peuvent avoir un effet significatif sur le total d'espace de swap disponible.

Chaque variable utilise 20 octets dans les segments mémoire, qui grossissent par paquet d'1 K ou plus. Les variables mémoire ont également besoin d'information de symbole pour être conservées pendant l'exécution. La conversion des variables PUBLIC et PRIVATE en LOCAL et STATIC peuvent réduire les besoins de mémoire pour la plupart des applications.

Table des symboles

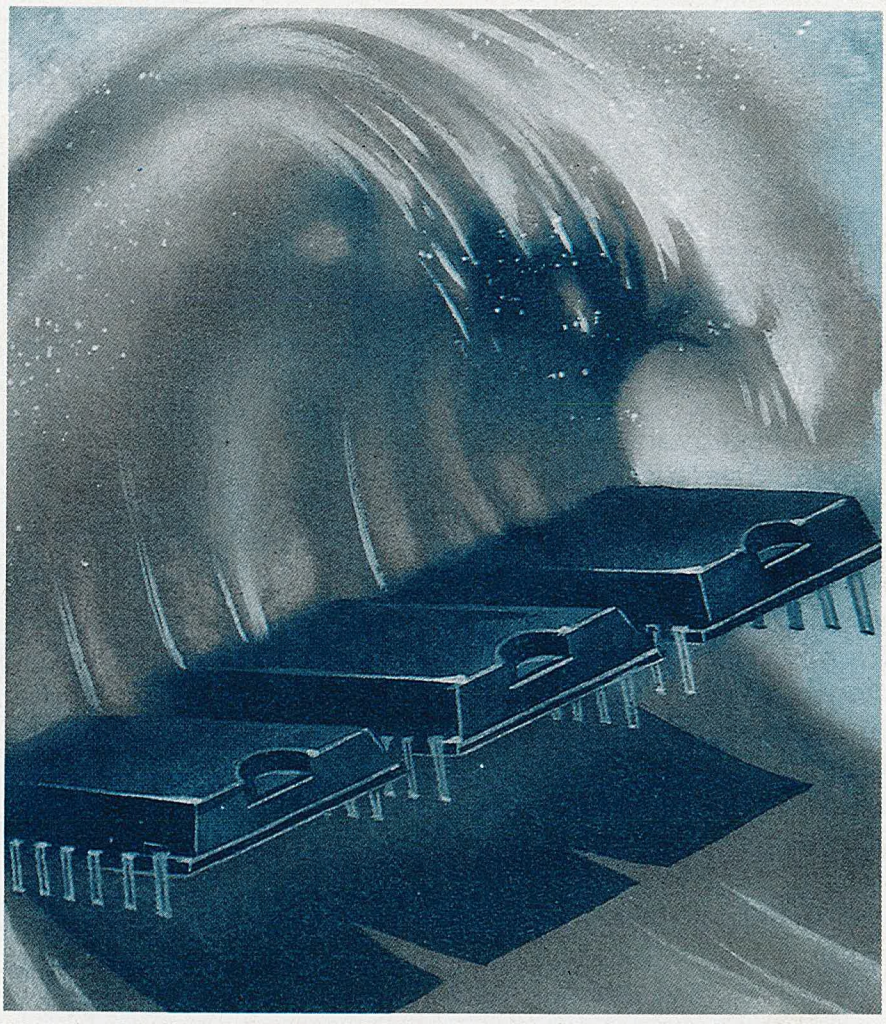
La plupart des informations sur les symboles sont contenues dans le .EXE. D'autres symboles sont créés dynamiquement à l'exécution. La mémoire pour la table des symboles et les informations qui s'y rapportent est utilisée par l'intermédiaire du gestionnaire de mémoire fixe. Les symboles sont placés en blocs et ne fragmentent pas la mémoire. Néanmoins, la création dynamique excessive de symboles (par exemple un grand nombre de variables mémoire créés par l'intermédiaire des macros) produit un accroissement de la taille des tables de symbole et une réduction de l'espace de swap.

Le Extend System

Le Système Extend est un ensemble de fonctions qui sont utilisées par les langages C et Assembleur pour créer des interfaces avec les programmes Clipper. Le Système Extend comprend des fonctions d'allocation de la mémoire, _xalloc() et _xgrab(), qui permettent aux programmes en C de récupérer la mémoire gérée par Clipper. Ces fonctions font appel au gestionnaire de mémoire fixe.

Le système Extend permet également aux programmes écrits en C, l'accès aux valeurs des données Clipper. La fonction _parc() du Système Extend, utilisée pour inspecter les valeurs caractère passées depuis Clipper, retourne une adresse en mémoire réelle. Comme les valeurs caractère sont stockées en mémoire virtuelle sous le SVOS, le Système Extend bloque automatiquement les segments de VM appropriés en réponse aux requêtes de la fonction _parc(). Ces segments sont ensuite automatiquement débloqués quand les fonctions C redonnent le contrôle à Clipper. Ils peuvent également être débloqués ''manuellement'' à partir des fonctions C.

Un conflit peut survenir si une fonction C bloque un segment de VM par l'intermédiaire de _parc() et essaye



ensuite de réserver une part importante de la mémoire en utilisant les gestionnaires de mémoire du Système Extend. Si le segment est bloqué tout près de la fin de l'espace de swap, le gestionnaire de mémoire fixe est incapable de bloquer ou d'agrandir un nouveau segment pour satisfaire à la demande d'occupation.
Si cela se produit, un nouveau segment fixe est créé et bloqué plus haut en mémoire. Ceci réduit l'efficacité du VMM qui n'est pas autorisé à déplacer des segments fixes. Dans les cas graves vous obtiendrez le triste message ''swap space exhausted'' parce que l'espace de swap est devenu trop encombré de segments fixes. Pour éviter ce genre de situation, le VMM réserve une ''zone de sécurité'' tout en bas de l'espace de swap. Les demandes de la fonction _parc() ne peuvent pas bloquer de segments dans cette zone.

La fonction _xunlock(), appelable depuis une fonction en C ou en Assembleur, permet de déverrouiller les segments gérés par le SVOS quand des quantités importantes de mémoire doivent être occupées suite à un appel de _parc(). Attention : tous les pointeurs renvoyés par des appels précédents de _parc() deviennent invalides. Pour avoir accès de nouveau à une valeur caractère Clipper, il faut absolument effectuer un nouvel appel à _parc() après avoir appelé _xunlock(), pour obtenir un nouveau pointeur valide.
Ci-dessous, une petite fonction, appelable depuis le debugger ou depuis un programme, pour surveiller cette fameuse mémoire.

Dans le prochain article, nous referons un tour d'horizon sur les macros, avec un bref rappel des potentialités que possédait déjà Clipper 87 et surtout nous examinerons les nouvelles possibilités de Clipper 5 en la matière, grâce à l'emploi des codes-blocks.

Nous parlerons également de l'avenir de Clipper 5 : réelle programmation objet et environnement graphique. Tout ceci est d'ores est déjà possible avec pour la Programmation Orientée Objet, la librairie Class(y), et pour l'interface graphique, les librairies Provisious Windows, distribuée par LDQS, et VI : (Visual Interface) distribuée par PC Technologie. Ces deux produits permettent un look Windows sous Dos, avec multifenêtrage et programmation événementielle, icônes, Souris et tout et tout ! Le compte rendu complet dans le prochain numéro.

Je reviens rapidement sur la gestion des erreurs et l'article paru dans le n° 72, de La Revue de l'Utilisateur PC, Septembre 1991. Il était très complet, un peu trop tout de même. Je reprends ci-dessous une explication plus courte et plus limpide.

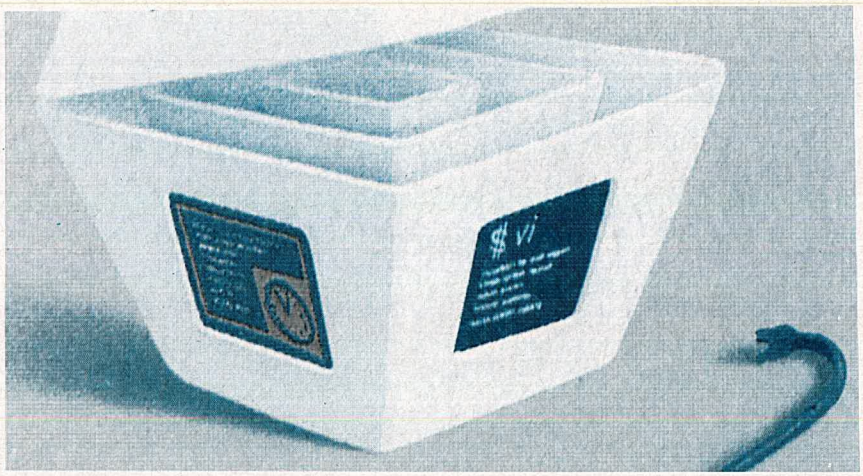
```
 function DispMem(n)
/*
Affiche la mémoire disponible
renvoie .T. si elle descend en dessous d'un certain
niveau (option)
Exemples de ligne de commande pour le débugger:

? dispmem()       // Affiche la boîte d'état de la mémoire
pt dispmem(125)   // S'arrête quand la mémoire descend en
dessous de 125 k. Parmi les possibilités nouvelles de Clipper
5.01, sachez que memory(3) et memory(4) fonctionnent très
bien pour l'EMS et la zone du VMM.

*/
local cClr,nOldcurseur,cMemscr
if n == nil
  cClr := setcolor("W+/R")      //récupération et assignation
  nOldcurseur := setcursor(0)   // dans la foulée
  cMemscr := Savescreen(10,35,16,60)
  Scroll(10,35,16,60)
  Dispbox(10,35,16,60,2)
  @ 11, 36 say "Etat de la mémoire    KO"
  @ 12, 36 say "------------------------"
  @ 13, 36 say "Total disponible   " +str(memory(0), 4)
  @ 14, 36 say "Bloc le plus grand " +str(memory(1), 4)
  @ 15, 36 say "Zone de RUN        " +str(memory(2), 4)
  inkey(0)
  setcolor(cClr)
  setcursor(nOldcurseur)
  Restscreen(10,35,16,60,cMemscr)
  n := 0
endif
 return (memory(0) < n)

 // Exemple de fonction appelable depuis le debugger à partir
//  de la fenêtre de commande, celle du bas !!!
```

```
// Démonstration de la récupération des erreurs de runtime
Function Main                  // prog principal
LOCAL MesErreur, row := 10  // message et ligne d'affichage

// déclaration d'un code block PUBLIC au début du
// prg principal et passage d'un message au cargo de l'objet.
PUBLIC bErreur := { |e| e:cargo:="On est planté", MonErreur(e) }

// passage de ce bloc à ErrorBlock(), pour remplacer
// celui passé par défaut par Clipper.
// Maintenant c'est le notre qui est actif.
ErrorBlock(bErreur)
cls
While .T.

   // Boucle Begin Sequence
   BEGIN SEQUENCE

   // j'espère que vous n'avez pas un fichier de ce nom là !!!
   USE testxyz  // censé ne pas exister

   // clause recover using pour récupérer
   // le contenu du message renvoyé par Break()
   RECOVER USING MesErreur

   // en cas d'erreur d'exécution, on se retrouve SOUS
   // l'instruction RECOVER. A partir de là, on peut
   // intercepter l'erreur et la traiter à sa convenance.
   Do Case
      Case MesErreur == "LOOP"
      @ row,20 Say "Fichier non trouvé, on recommence"
      Inkey(5)
      ++row
      Loop
      Case MesErreur == "EXIT"
      @ row,20 Say "Fichier non trouvé, on quitte"
      Inkey(5)
      exit
   EndCase
   END
EndDo
cls
return nil

Function MonErreur(e)
// le paramètre "e" contient l'objet erreur créé par
// ErrorNew() automatiquement. Vous pouvez, bien entendu
// récupérer toutes les variables d'instances de cet objet.
STATIC counter := 0
Static row := 5
@ Maxrow(),row say e:cargo
inkey(0)
counter++
row:=row+20
break(IF(counter < 3,"LOOP","EXIT"))
return nil
```

Conversion d'images graphiques : du format Bit-map au format PCX et vice versa

Travailler en mode graphique est devenu réellement une "mode" suivie par tout le monde à un tel point que la plupart des logiciels et des applications se convertissent ou s'intègrent sous l'environnement WINDOWS qui, il faut le reconnaître, a rendu le dialogue entre la machine et l'utilisateur agréable, confortable et convivial.

Salah RIMANI



Alors nous essayons de rattraper le convoi en créant une interface entre WINDOWS et TURBO PASCAL. Ce dernier représente un langage de programmation bien structuré et simple mais, dont l'efficacité reste assez limitée. A mainte reprise et pour mettre au point un programme, on est amené à créer nous-mêmes plusieurs routines. Parmi celles-ci, nous en avons choisi deux.

1) Une procédure qui permet de traduire une image créée en TURBO PASCAL par une autre lisible par Paint Brush (sous WINDOWS 3.00).

2) Une deuxième procédure qui pourrait être complémentaire à la première. Celle-ci traite le sens inverse, conversion d'une image au format PCX de Paint Brush en une image Bit-map de TURBO PASCAL.

L'élaboration de cette interface est le fruit d'une étude minutieuse des deux formats.

A) Format Bit-map de TURBO PASCAL : pour comprendre la structure de ce format d'image généré par la routine GETIMAGE de TURBO PASCAL, nous nous sommes servis essentiellement du fameux utilitaire PCTOOLS (version 4.30).

En effet, une image Bit-map selon la conception de BORLAND est constituée fondamentalement de deux parties.

— Une en-tête de 4 octets qui mémorise la largeur et la hauteur de l'image. Car celle-ci est toujours inscrite dans un cadre rectangulaire. Les deux premiers octets représentent la largeur et les octets suivants la hauteur. A noter que la norme suivant laquelle est codée un entier ou un mot, est celle adoptée par INTEL (l'octet du poids faible suivi de l'octet du poids fort).

— Une description détaillée de l'image ligne par ligne et au sein de chaque ligne plan par plan. La seule particularité à noter à cet égard réside dans le fait que les plans de bits se succèdent de la façon suivante.

{xxxxxxxxxxxxxx.....xxxxx} plan de bit 0 (Intensité)
{xxxxxxxxxxxxxx.....xxxxx} plan de bit 1 (Rouge)
{xxxxxxxxxxxxxx.....xxxxx} plan de bit 2 (Vert)
{xxxxxxxxxxxxxx.....xxxxx} plan de bit 3 (Bleu)

Le bloc des données est terminé par deux octets "bidons", ne nous demandez pas pourquoi, ce qui augmente de deux la taille mémoire requise par une image. Théoriquement, cette taille se calcule comme suit : L * H * N. Sachant que L constitue le nombre d'octets nécessaires pour mémoriser un seul plan de bits sur une ligne d'image ; H est la hauteur de l'image et N représente le nombre de plans de bits qui est égal à 4 pour une image 16 couleurs.

La fonction IMAGESIZE de TURBO PASCAL génère toujours une valeur supérieure de 6 octets à celle que donne la formule. On sait que 4 parmi ces 6 octets sont réservés à l'en-tête (2 pour la largeur et 2 pour la hauteur), les deux autres sont situés à la fin de l'image et sont a priori inutiles.

B) Suivant les cas, un fichier image au format PCX de ZSoft's est constitué de deux ou trois parties :

— une en-tête,

— les données de l'image proprement dite,

— et une palette de 256 couleurs pour certains fichiers.

Aucun renseignement n'est fourni sur le nombre de plans de bits qui est à notre avis une information capitale dans la manipulation des images. Inversement, le format PCX de WINDOWS est un format relativement complexe et qui procure beaucoup d'informations dont quelques-unes sont, à la limite, inutiles.

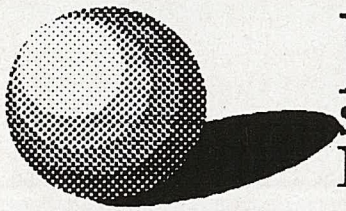
Pour de plus amples détails sur la structure d'un fichier PCX, on vous renvoit à l'article de Jean-Michel SPINER paru dans La Revue de l'Utilisateur PC n° 72, Septembre 1991. On se contente ici de rappeler que la seconde partie est réservée aux données de l'image elle-même et que ces données sont compressées par la méthode du RLC (Run Length Coding) qui a été légèrement modifiée.

Une fois décompressées, les données se suivent ligne par ligne et au sein de chaque ligne les plans de bits se succèdent tout en sachant qu'à l'inverse de TURBO PASCAL, Paint Brush place le plan de bits d'intensité en quatrième position.

OFFRE DE LANCEMENT

jusqu'au 31/03/92

le SHAREWARE du PROFESSIONNEL



PRO. Shareware

SARL au capital de 50000 F

BP 11 71120 CHAROLLES

1 disquette GRATUITE pour chaque commande

Procurez-vous de nombreux outils de programmations dans de nombreux langages, ainsi que des utilitaires et jeux du DOMAINE PUBLIC ou SHAREWARE internationaux au prix de 30 frs la disquette

envoyer à PROSHAREWARE BP 11 71120 CHAROLLES

Demande de CATALOGUE

Nom : Prénom :

Adresse :

Code Postal : Ville :

PC0292

CONVERSION D'IMAGE GRAPHIQUE :

du format Bit-map au format PCX et vis versa.

Rimani Salah Janvier 1992

```
USES crt, graph;

CONST
 Enter   = #13;
 Esc     = #27;
 Nul     = #00;
 FHaut   = 'H';
 FBas    = 'P';
 FDroite = 'M';
 FGauche = 'K';
 FDebut  = 'G';
 FFin    = 'O';
 FPgDn   = 'Q';
 FPgUp   = 'I';
 ToucheDepl = [FHaut, FBas, FDroite, FGauche, FDebut, FFin, FPgDn, FPgUp];
 Puce       = 5;
 MaxBit     = 255;
 Drapeau    = 192;
 TaillePal  = 769;
 TailleEntete = 128;
 MaxOctetsLgn = 2054;
 MaxBytes   = 64016;
 MaxBits    = 65520;
 MaxOctLgn  : WORD = 326;
 Palette16 : ARRAY[1..48] OF BYTE =
        { R   G   B }
        ( 0,  0,  0,
        128,  0,  0,
          0,128,  0,
        128,128,  0,
          0,  0,128,
        128,  0,128,
          0,128,128,
        128,128,128,
        192,192,192,
        255,  0,  0,
          0,255,  0,
        255,255,  0,
          0,  0,255,
        255,  0,255,
          0,255,255,
        255,255,255);

TYPE
 BufferImage = ^ScanImage;
 ScanImage = ARRAY[1..MaxBytes] OF BYTE;
 BufferPalette = ARRAY[1..TaillePal] OF BYTE;
 BufferScanLine = ARRAY[1..MaxOctetsLgn] OF BYTE;
 EntetePCX = RECORD
           Griffe   : BYTE;
           Version  : BYTE;
           Codage   : BYTE;
           BitsPixel : BYTE;
           X1       : WORD;
           Y1       : WORD;
           X2       : WORD;
           Y2       : WORD;
           ResH     : WORD;
           ResV     : WORD;
           Palette  : ARRAY[1..48] OF BYTE;
           Bidon    : BYTE;
           NbPlans  : BYTE;
           OctetsLgn : WORD;
           TypePal  : WORD;
           XDpi     : WORD;
           YDpi     : WORD;
           Toto     : ARRAY[1..54] OF BYTE;
         END;
 TypeImageGraphique = (Mono, Coul4, Coul16, Coul256);
 CoordonneesImages = RECORD
                Y1, Haut, Size : WORD;
              END;

VAR
 X1Loc, Y1Loc, X2Loc, Y2Loc, MaxX, MaxY : INTEGER;
 BlocImage : ARRAY[1..4] OF CoordonneesImages;
 EnteteBidon : STRING[TailleEntete];
 TypeImage : TypeImageGraphique;
 PalCour : palettetype;
 Pal : BufferPalette;
 Entete : EntetePCX;
 Charge : BOOLEAN;
 FSize : LONGINT;
 Marq : POINTER;

{Cette routine permet de de déplacer le cadre de l'image n'importe où à
 l'écran, et de choisir la position où l'image devrait être affichée.}
PROCEDURE FixeCadreImage(VAR XX1, YY1, XX2, YY2 : INTEGER;
                         VAR DebX, DebY : INTEGER);
CONST
 Incr = 8;

VAR
 Ch : CHAR;
 Fen : viewporttype;
 X1, Y1, DX, DY : INTEGER;

 {Sachez qu'en Turbo Pascal, on ne peut pas manipuler un seul bloc d'image
  dont la taille dépasse les 64 K. Cette procédure a été crée pour combler
  cette lacune.}
 PROCEDURE DiviseImage;
 VAR
  YBar, Ys, I : WORD;

 BEGIN
  fillchar(BlocImage, sizeof(BlocImage), 0);
  I := 0;
  YBar := 0;
  Ys := YY1;
  WHILE Ys + YBar < YY2 DO
   BEGIN
    WHILE (imagesize(XX1, Ys, XX2, Ys + YBar) < MaxBytes - MaxOctLgn)
          AND (Ys + YBar < YY2) DO inc(YBar);
    inc(I);
    BlocImage[I].Y1 := Ys;
    BlocImage[I].Haut := YBar;
    BlocImage[I].Size := imagesize(XX1, Ys, XX2, Ys + YBar) - 6;
    inc(Ys, succ(YBar));
    YBar := 0;
   END;
 END;

 {Cette procédure visualise la totalité de la surface de l'image par un
  motif sous forme de filet.}
 PROCEDURE Filet;
 BEGIN
  setfillstyle(8, 13);
  bar(X1, Y1, X1 + DX, Y1 + DY);
  rectangle(X1, Y1, X1 + DX, Y1 + DY);
  line(X1 + DX + 1, Y1 + 1, X1 + DX + 1, Y1 + DY + 1);
  line(X1 + 1, Y1 + DY + 1, X1 + DX + 1, Y1 + DY + 1);
 END;
```

```
BEGIN
 getviewsettings(Fen);
 rectangle(X1Loc - 2, Y1Loc - 2, X2Loc + 2, Y2Loc + 2);
 setviewport(X1Loc, Y1Loc, X2Loc, Y2Loc, TRUE);
 WITH Entete DO
  BEGIN
   DX := X2 - X1;
   DY := Y2 - Y1;
  END;
 X1 := 0;
 Y1 := 0;
 setlinestyle(0, 0, 1);
 setcolor(7);
 Filet;
 REPEAT
  Ch := readkey;
  IF Ch = Nul THEN
   BEGIN
    Ch := readkey;
    IF Ch IN ToucheDepl THEN
     BEGIN
      setfillstyle(1, 0);
      bar(X1, Y1, X1 + DX + 1, Y1 + DY + 1);
      CASE Ch OF
       FHaut : dec(Y1, Incr);
       FBas  : inc(Y1, Incr);
       FDroite : inc(X1, Incr);
       FGauche : dec(X1, Incr);
       FDebut : BEGIN
                 dec(X1, Incr);
                 dec(Y1, Incr);
                END;
       FPgUp : BEGIN
                 inc(X1, Incr);
                 dec(Y1, Incr);
                END;
       FPgDn : BEGIN
                 inc(X1, Incr);
                 inc(Y1, Incr);
                END;
       FFin  : BEGIN
                 dec(X1, Incr);
                 inc(Y1, Incr);
                END;
      END;
      IF Y1 < -pred(DY) THEN Y1 := -pred(DY);
      IF Y1 > pred(Y2Loc - Y1Loc) THEN Y1 := pred(Y2Loc - Y1Loc);
      IF X1 < -pred(DX) THEN X1 := -pred(DX);
      IF X1 > pred(X2Loc - X1Loc) THEN X1 := pred(X2Loc - X1Loc);
      Filet;
     END;
   END;
 UNTIL (Ch = Enter) OR (Ch = Esc);
 clearviewport;

 DebX := 0;
 DebY := 0;

 IF Ch = Enter THEN
  BEGIN
   XX1 := X1Loc + X1;
   YY1 := Y1Loc + Y1;
   IF X1 < 1 THEN
    BEGIN
     XX1  := X1Loc;
     DebX := abs(X1);
    END;
   IF Y1 < 1 THEN
    BEGIN
     YY1  := Y1Loc;
     DebY := abs(Y1);
    END;
   END
  ELSE
   BEGIN
    inc(XX1, X1Loc);
    inc(YY1, Y1Loc);
   END;

  XX2  := XX1 + (DX - DebX);
  YY2  := YY1 + (DY - DebY);
  IF XX2 > X2Loc THEN
   XX2 := X2Loc;
  IF YY2 > Y2Loc THEN
   YY2 := Y2Loc;

  WITH Fen DO
   setviewport(X1, Y1, X2, Y2, clip);
  DiviseImage;
 END;

{Initialise l'écran en mode graphique, ainsi que d'autres variables.}
PROCEDURE Initialise;
VAR
 XAsp, YAsp : WORD;
 CodeErreur, Gd, Gm : INTEGER;

BEGIN
 Gd := detect;
 initgraph(Gd, Gm, 'c:\tp\bgi');
 CodeErreur := graphresult;
 IF CodeErreur <> grok THEN
  BEGIN
   writeln('Erreur Graphique :');
   writeln(GraphErrorMsg(CodeErreur));
   writeln('Interruption immédiate du programme...');
   Halt(1);
  END;

 fillchar(Entete, sizeof(Entete), 0);
 fillchar(Pal, sizeof(Pal), 0);
 getpalette(PalCour);
 MaxX := getmaxx;
 MaxY := getmaxy;
 GetAspectRatio(XAsp, YAsp);
 X1Loc := 7; Y1Loc := round(X1Loc * (XAsp / YAsp));
 X2Loc := MaxX - 8;  Y2Loc := MaxY - 22;

 setfillstyle(1, 12);
 bar(0, 0, MaxX, MaxY);
 setfillstyle(1, 0);
 bar(pred(X1Loc), pred(Y1Loc), succ(X2Loc), succ(Y2Loc));
END;

{Cree une entête pour un format PCX.}
PROCEDURE CreeEntetePCX(XGauche, YHaut, XDroite, YBas : INTEGER);
BEGIN
 WITH Entete DO
  BEGIN
   Griffe   := $0A;
   Version  := $05;
   Codage   := $01;
   BitsPixel := $01;
   CASE PalCour.Size OF
     4 : BitsPixel := $02;
   255 : BitsPixel := $08;
   END;
```

```
    X1  := XGauche;
    Y1  := YHaut;
    X2  := XDroite;
    Y2  := YBas;
    ResH := MaxX;
    ResV := MaxY;
    move(Palette16, Palette, sizeof(Palette));
    Bidon  := $A4;
    NbPlans := $01;
    CASE PalCour.Size OF
     2..4 : NbPlans := $02;
       16 : NbPlans := $04;
    END;
    OctetsLgn := (XDroite - XGauche) SHR 3 + 1;
    IF PalCour.Size > 16 THEN OctetsLgn := XDroite - XGauche;
    Typepal := 1 - Mem[$40:$49] DIV 7;
    XDpi   := MaxX;
    YDpi   := MaxY;
    fillchar(Toto, sizeof(Toto), 0);
  END;
END;

{Affiche un message en bas de l'écran.}
PROCEDURE AfficheMessage(Message : STRING; CF, CT : WORD);
BEGIN
 setfillstyle(1, CF);
 bar(70, MaxY - 18, MaxX - 70, MaxY);
 setcolor(14);
 rectangle(71, MaxY - 17, MaxX - 71, pred(MaxY));
 setcolor(CT);
 settextstyle(0, 0, 1);
 settextjustify(centertext, toptext);
 outtextxy(MaxX DIV 2, MaxY - 12, Message);
END;

{Emet un signal sonore}
PROCEDURE UnBip;
BEGIN
 sound(100);
 delay(70);
 sound(150);
 delay(100);
 sound(200);
 nosound;
END;


{Affiche un message d'erreur}
PROCEDURE Erreur(Code : byte);
BEGIN
 UnBip;
 CASE Code OF
  1 : AfficheMessage('Mémoire insuffisante !', 4, 14);
  2 : AfficheMessage('Erreur en lecture !', 12, 15);
  3 : AfficheMessage('Erreur en écriture !', 12, 15);
  4 : AfficheMessage('Format inconnu !', 4, 15);
 END;
END;

{Décode l'entête et eventuellement la fin d'un fichier au format
 PCX. Et, définit le type de l'image graphique.}
PROCEDURE DecodeEnteteFinPCX(VAR F : FILE);
VAR
 BlocLu : WORD;

BEGIN
 FSize := filesize(F);
 {$i-} blockread(F, Entete, TailleEntete, BlocLu); {$i+}
IF (ioresult <> 0) OR (BlocLu <> TailleEntete) THEN
 BEGIN
  Erreur(2);
  exit;
 END;

IF filesize(F) > TaillePal THEN
 BEGIN
  seek(F, filesize(F) - TaillePal);
  {$i-} blockread(F, Pal, TaillePal, BlocLu); {$i+}
  IF (ioresult <> 0) OR (BlocLu <> TaillePal) THEN
   BEGIN
    Erreur(2);
    exit;
   END;
 END;

WITH Entete DO
 BEGIN
  CASE BitsPixel OF
   1 : BEGIN
       IF NbPlans = 1 THEN TypeImage := Mono;   {couleurs 2, surface 1}
       IF NbPlans = 4 THEN TypeImage := Coul16; {couleurs 16, surface 4 ega vga}
       END;
   2 : IF NbPlans = 1 THEN TypeImage := Coul4;   {couleurs 4, surface 1 cga cgac1 300x200}
   8 : IF NbPlans = 1 THEN TypeImage := Coul256; {couleurs 256, surface 1}
  END;
 END;
{$i-} close(F); {$i+}
END;

{Teste si la palette des couleurs obeillie à la norme
 Red Green Blue de windows PCX.}
FUNCTION TestPalette : BOOLEAN;
VAR
 I : SHORTINT;

BEGIN
 TestPalette := FALSE;
 FOR I := 1 TO 48 DO
  IF Entete.Palette[I] <> Palette16[I] THEN
   TestPalette := TRUE;
END;

{Traduit le format PCX de Windows au format Bit-Map de Turbo Pascal et donne
 la possiblité de définir le mode d'affichage de l'image :
 CopyPut, XorPut, NotPut, OrPut et AndPut.}
FUNCTION WindoPCX_BMP(NomF : STRING; ModeAffichage : WORD) : BOOLEAN;

VAR
 F : FILE;
 Fen : viewporttype;
 Octet, Nbre : BYTE;
 Fic, FMap : BufferImage;
 PalBidon, PileVide : BOOLEAN;
 Buffer, BitMap : BufferScanLine;
 LongDonnee, SizeCumul : LONGINT;
 YBloc, YBar, DebY, DebX : INTEGER;
 BitsLgn, PtrS, Bl, BlocLu, BlocEcr, Cumul, SegX : WORD;
 NbPlans1, Large, Haut, XGauche, YHaut, XDroite, YBas : INTEGER;


 PROCEDURE Empile(Ligne : BufferScanLine);
 BEGIN
  move(Ligne, FMap^[Puce + SizeCumul], BitsLgn);
  inc(SizeCumul, BitsLgn);
  PileVide := FALSE;
 END;
```

```
  PROCEDURE Depile;
  BEGIN
   move(BlocImage[YBloc].Haut, FMap^[3], 2);
   putimage(XGauche, BlocImage[YBloc].Y1, FMap^, ModeAffichage);
   inc(YBloc);
   SizeCumul := 0;
   PileVide := TRUE;
  END;

  PROCEDURE TraiteLigne;
  VAR
   I, J : INTEGER;
   Pixel : BYTE;

  BEGIN
   WITH Entete DO
    BEGIN
     IF TypeImage = Coul256 THEN
      BEGIN
       FOR I := 1 TO Large DO
        BEGIN
         Pixel := GetPixel(XGauche + pred(I), YHaut);
         CASE ModeAffichage OF
          CopyPut : Pixel := Buffer[I + DebX];
          AndPut  : Pixel := Buffer[I + DebX] AND Pixel;
          OrPut   : Pixel := Buffer[I + DebX] OR Pixel;
          XorPut  : Pixel := Buffer[I + DebX] XOR Pixel;
          NotPut  : Pixel := $FF - Buffer[I + DebX];
         END;

         putpixel(XGauche + pred(I), YHaut, Pixel);
        END;
      END
     ELSE
      BEGIN
       IF NOT PalBidon THEN
        BEGIN
         move(Buffer[DebX + OctetsLgn * pred(NbPlans)],
           BitMap, SegX);
         FOR I := 1 TO NbPlans - 1 DO
          move(Buffer[DebX + (I - 1) * OctetsLgn],
            BitMap[1 + I * SegX], SegX * pred(NbPlans));
        END
       ELSE
        FOR I := 1 TO NbPlans DO
         move(Buffer[DebX + (NbPlans - I) * OctetsLgn],
           Bitmap[1 + (I - 1) * SegX], SegX);
       IF (PalCour.Size = 16) AND
        ((TypeImage = Mono) OR (TypeImage = Coul4)) THEN
        FOR I := 1 TO NbPlans1 - 1 DO
         FOR J := 0 TO SegX - 1 DO
          IF BitMap[1 + J + I * SegX] <> 255 THEN
           BitMap[1 + J + I * SegX] := 255;

       IF SizeCumul >= BlocImage[YBloc].Size THEN Depile;
       Empile(BitMap);
      END;
     inc(YHaut);
    END;
  END;

BEGIN
 WindoPCX_BMP := FALSE;
 assign(F, NomF);
 {$i-} reset(F, 1); {$i+}
 IF Ioresult <> 0 THEN
  BEGIN
   Erreur(2);
   exit;
  END;
 Fic := NIL;
 FMap := NIL;
 SizeCumul := 0;
 YBloc := 1;
 PileVide := TRUE;
 getmem(FMap, MaxBytes);
 IF FMap = NIL THEN
  BEGIN
   Erreur(1);
   exit;
  END;
 fillchar(FMap^, MaxBytes, 0);
 getviewsettings(Fen);
 setviewport(0, 0, MaxX, MaxY, TRUE);

 DecodeEnteteFinPCX(F);

 PalBidon := TestPalette;
 IF PalBidon THEN
  BEGIN
   UnBip;
   IF ((Entete.Griffe <> $0A) AND (Entete.Griffe <> $0C))
    OR (FSize < TailleEntete) THEN
    BEGIN
     Erreur(4);
     delay(4500);
     exit;
    END
   ELSE
    AfficheMessage('Palette des couleurs incompatible !', 4, 14);
   delay(4500);
  END;

 AfficheMessage('Flèches=Déplacement du cadre. ESC=Annulation. ENTER=Fixation', 1, 15);

 XGauche := 0;
 YHaut := 0;

 FixeCadreImage(XGauche, YHaut, XDroite, YBas, DebX, DebY);

 reset(F, 1);

 LongDonnee := FSize;
 IF TypeImage = Coul256 THEN
  LongDonnee := FSize - TaillePal;

 {$i-} blockread(F, EnteteBidon, TailleEntete, BlocLu); {$i+}
 IF (ioresult <> 0) OR (BlocLu <> TailleEntete) THEN
  BEGIN
   Erreur(2);
   exit;
  END;

 getmem(Fic, sizeof(Fic));
 IF Fic = NIL THEN
  BEGIN
   Erreur(1);
   exit;
  END;

 WITH Entete DO
  BEGIN
   MaxOctLgn := NbPlans * OctetsLgn;
   NbPlans1 := NbPlans;
  END;
 IF (PalCour.Size = 16) AND
  ((TypeImage = Mono) OR (TypeImage = Coul4)) THEN NbPlans1 := 4;

 IF TypeImage <> Coul256 THEN
  DebX := DebX SHR 3 + 1;
```

```
Large := XDroite - XGauche;

IF TypeImage = Coul256 THEN
 inc(Large);
SegX := Large SHR 3 + 1;
BitsLgn := SegX * NbPlans1;
move(Large, FMap^, 2);

YBar := 0;
Haut := 0;

AfficheMessage('Patience, chargement en cours...', 3, 14);

Cumul := 1;
WHILE (filepos(F) < LongDonnee) AND (YHaut <= YBas) DO
 BEGIN
  fillchar(fic^, sizeof(fic), 0);
  {$i-} blockread(F, Fic^, MaxBytes - 1, BlocLu); {$i+}
  IF ioresult <> 0 THEN
   BEGIN
    Erreur(2);
    exit;
   END;
  PtrS := 1;
  WHILE (PtrS <= BlocLu) AND (YHaut <= YBas) DO
   BEGIN
    Octet := Fic^[PtrS];
    IF (Octet >= Drapeau) AND (PtrS <= BlocLu) THEN
     BEGIN
      Nbre := Octet - Drapeau;
      inc(PtrS);
      IF PtrS > BlocLu THEN
       BEGIN
        {$i-} blockread(F, Fic^[PtrS], 1, Bl); {$i+}
        IF (ioresult <> 0) OR (Bl <> 1) THEN
         BEGIN
          Erreur(2);
          exit;
         END;
       END;
      Octet := Fic^[PtrS];
     END
    ELSE Nbre := 1;
    IF Cumul > MaxOctLgn THEN
     BEGIN
      IF YBar >= DebY THEN
       TraiteLigne;
      inc(YBar);
      Cumul := 1;
     END;
    fillchar(Buffer[Cumul], Nbre, Octet);
    inc(Cumul, Nbre);
    inc(PtrS);
   END;
 END;

 IF (Cumul > 1) AND (YHaut <= YBas) THEN
  TraiteLigne;
 IF NOT PileVide THEN Depile;
 {$i-} close(F); {$i+}
 FOR YBar := 1 TO length(NomF) DO
  NomF[YBar] := upcase(NomF[YBar]);

 AfficheMessage(NomF, 4, 10);
 UnBip;
 WindolPCX_BMP := TRUE;
END;
```

```
{Cette routine se charge de traduire le format Bit-Map de Turbo Pascal
 au format PCX de Paint Brush.}
PROCEDURE TurboBMP_PCX(X1, Y1, X2, Y2 : INTEGER; NomF : STRING);
VAR
 F : FILE;
 YBar : INTEGER;
 NPlans : SHORTINT;
 Fen : viewporttype;
 BS, BD : BufferScanLine;
 NOctPlan, BlocEcr, Compress : WORD;

 {Fonction permettant de coder l'image suivant la méthode RLC.}
 FUNCTION EncodeBMP_PCX(VAR BS, BD : BufferScanLine) : WORD;
 VAR
  PtrS, PtrD : WORD;
  Octet, Nbre : BYTE;

 BEGIN
  PtrS := 1;
  PtrD := 0;
  WHILE PtrS <= MaxOctLgn DO
   BEGIN
    Octet := BS[PtrS];
    IF (PtrS < MaxOctLgn) AND (Octet = BS[succ(PtrS)]) THEN
     BEGIN
      Nbre := Drapeau + 2;
      inc(PtrS);
      WHILE (Octet = BS[succ(PtrS)]) AND
          (Nbre < MaxBit) AND (PtrS < MaxOctLgn) DO
       BEGIN
        inc(Nbre);
        inc(PtrS);
       END;
      inc(PtrD);
      BD[PtrD] := Nbre;
      inc(PtrD);
      BD[PtrD] := Octet;
     END
    ELSE
     BEGIN
      IF Octet >= Drapeau THEN
       BEGIN
        inc(PtrD);
        BD[PtrD] := succ(Drapeau);
        inc(PtrD);
        BD[PtrD] := Octet;
       END
      ELSE
       BEGIN
        inc(PtrD);
        BD[PtrD] := Octet;
       END;
     END;
    inc(PtrS);
   END;
  EncodeBMP_PCX := PtrD;
 END;

BEGIN
 AfficheMessage('Sauvegarde en cours...', 3, 5);
 assign(F, NomF);
 {$i-} rewrite(F, 1); {$i+}
 IF Ioresult <> 0 THEN
  BEGIN
   Erreur(3);
   exit;
  END;
 getviewsettings(Fen);
 setviewport(0, 0, MaxX, MaxY, TRUE);
 inc(X1, Fen.x1);
```

```
    inc(X2, Fen.x1);
    inc(Y1, Fen.y1);
    inc(Y2, Fen.y1);
    CreeEntetePCX(X1, Y1, X2, Y2);
    {$i-} blockwrite(F, Entete, TailleEntete, BlocEcr); {$i+}
    IF (ioresult <> 0) OR (BlocEcr <> TailleEntete) THEN
      BEGIN
        Erreur(3);
        exit;
      END;

    NPlans  := Entete.NbPlans;
    NOctPlan := Entete.OctetsLgn;
    MaxOctLgn := imagesize(X1, Y1, X2, Y1) - 6;

    YBar := Y1;
    WHILE YBar <= Y2 DO
      BEGIN
        getimage(X1, YBar, X2, YBar, BS);
        move(BS[Puce + NOctPlan], BD[1], NOctPlan * pred(NPlans));
        move(BS[Puce], BD[1 + pred(NPlans) * NOctPlan], NOctPlan);
        move(BD, BS, MaxOctLgn);
        Compress := EncodeBMP_PCX(BS, BD);
        {$i-} blockwrite(F, BD, Compress, BlocEcr); {$i+}
        IF (ioresult <> 0) OR (BlocEcr <> Compress) THEN
          BEGIN
            Erreur(3);
            exit;
          END;
        inc(YBar);
      END;
    {$i-} close(F); {$i+}
    AfficheMessage(NomF, 5, 10);
  END;

  {_____ programme principal _____}
  BEGIN
    Initialise;
    mark(Marq);
      Charge := WindofPCX_BMP('echecs.pcx', CopyPut);
      if Charge then
        BEGIN
          readln;
          TurboBMP_PCX(X1Loc, Y1Loc, X2Loc, Y2Loc,'toto.pcx');
        END;
    release(Marq);
    readln;
    closegraph;
  END.
```

BON DE COMMANDE

à retourner
à la Revue de l'Utilisateur PC
2, rue d'Amsterdam
75009 Paris

Nom : .....
Société : .....
Adresse : .....
Code postal : .....
Ville : .....
Tél. : .....

Montant : 21 F × = ...
24 F × = ...
(à partir du n° 45)
Total achat revues = ...

\+ Frais Poste = ...

Montant du chèque = ...

Règlement par chèque bancaire :
à l'ordre de SOFTin Communications : 2, rue d'Amsterdam, 75009 Paris
Frais poste pour 1 n° 7,50 F - 2 n° 14,00 F - 3/4 n° 18,00 F - 5/6 n° 25,00 F
(au-dessus par multiple)

POUR COMMANDER LES ANCIENS NUMEROS

Cochez les chiffres qui vous intéressent :

☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
☐ 6	☐ 7	☐ 8	☐ 9	☐ 10
☐ 11	☐ 12	☐ 13	☐ 14	☐ 15
☐ 16	☐ 17	☐ 18	☐ 19	☐ 20
☐ 21	☐ 22	☐ 23	☐ 24	☐ 25
☐ 26	☐ 27	☐ 28	☐ 29	☐ 30
☐ 31	☐ 32	☐ 33	☐ 34	☐ 35
☐ 36	☐ 37	☐ 38	☐ 39	☐ 40
☐ 41	☐ 42	☐ 43	☐ 44	☐ 45
☐ 46	☐ 47	☐ 48	☐ 49	☐ 50
☐ 51	☐ 52	☐ 53	☐ 54	☐ 55
☐ 56	☐ 57	☐ 58	☐ 59	☐ 60
☐ 61	☐ 62	☐ 63	☐ 64	☐ 65
☐ 66	☐ 67	☐ 68	☐ 69	☐ 70
☐ 71	☐ 72			

N° 9 - 17 - 18 - 19 - 20 - 22 - 23 - 34 - 35 - 36 - 37 épuisés

Optimisation des boucles en Assembleur

En dépit du développement croissant des langages de haut niveau, de la vogue des L4G (Langages de 4e Génération), l'Assembleur a conservé son rôle indispensable pour la réalisation d'applications, de modules, ou de routines à la durée d'exécution critiques, ou bien peu gourmandes en mémoire.

Paul FRANCESCHI

L'optimisation — en vitesse ou en mémoire — est en effet demeurée le domaine quasi exclusif de l'Assembleur.

Nombre de techniques permettant des gains appréciables en vitesse ou en occupation mémoire, ont déjà été explorées, et le présent article se propose de définir des techniques d'optimisation relatives à un domaine plus rarement abordé en Assembleur : celui des boucles.

Nous nous attacherons ainsi à étudier la méthode de programmation des boucles FOR...NEXT en Assembleur, en visant à les optimiser par rapport aux langages procéduraux classiques, tant au niveau de la vitesse d'exécution, que de la compacité du code généré.

Alors que les langages procéduraux proposent des formules générales, universelles, applicables en toutes circonstance et quels que soient les types d'objets manipulés (entiers, flottants, expressions numériques...), l'Assembleur nous permettra de définir des outils spécifiques et adaptés à un but particulier.

Le modèle de boucles que nous décrirons devra répondre aux spécifications suivantes :

— les valeurs limites de boucles (minimum et maximum) seront des entiers signés (compris entre – 32 267 et + 32 268) ;

— la lisibilité de la structure de boucle devra être au moins égale à celle d'un langage évolué ;

— l'imbrication des boucles devra être rendue possible. A cet égard, nous présenterons tout d'abord un type de boucle simple ; puis nous illustrerons le cas des boucles imbriquées ; bâties sur le même modèle.

Réalisation d'une boucle FOR...NEXT simple

Nous réaliserons ici l'équivalent d'une structure classique du type :

```
FOR (I = 1 ; I <= 10 ; I + +)
{
(...)
}
```

Dans cette structure caractéristique du C, les accolades définissent le début et la fin du traitement sur lequel porte la boucle, déclarée auparavant, avec ses extremums, et la variable sur laquelle elle porte : I. Dans un souci d'optimisation simultané de lisibilité, nous avons adopté la macro-instruction, qui réalise un traitement de boucle équivalent.

ITER_1 MACRO MIN1,MAX1,PAS1,SPG	Déclaration de Macro
LOCAL V1	
PUSH ITE_VAL1	
INIT ITE_VAL1,MIN1	Initialisation avec la valeur-minimum
V1:**MOV** BX,OFFSET SPG	Appel du traitement de boucle
CALL BX	
ADDI ITE_VAL1,PAS1,ITE_VAL1	Calcul de la valeur suivante
SI_INFEG ITE_VAL1,MAX1,V1	Teste si la valeur-maximum est atteinte
POP ITE_VAL1	
ENDM	

La macro ITER_1 réalise une boucle simple, et accepte en entrée 4 paramètres :

— une valeur minimum MIN1, pour la variable de boucle ;

— une valeur maximum MAX1;

— une valeur de pas de boucle PAS1 ;

— une adresse SPG, qui est un pointeur vers un traitement de boucle, simplement constitué par un sous-programme classique terminé par RET.

INIT et ADDI sont deux macros-instructions qui réalisent respectivement l'initialisation d'un mot et l'addition d'une valeur à un mot.

La variable sur laquelle porte la boucle est contenue dans ITE_VAL1, qui est une variable de type WORD.

L'implémentation d'une boucle avec cette technique demeure parfaitement lisible. Ainsi, la macro suivante détermine-t-elle une boucle de 100 à 1 000, par pas de 1, entraînant à chaque itération l'exécution du traitement situé en TRAITO.

```
ITER_1 100,1000,1,TRAITO
```

Réalisation de boucles imbriquées

Nous nous servirons de notre structure de boucle simple pour réaliser des boucles imbriquées, dont le besoin en programmation est évident.

Une double boucle imbriquée

La réalisation de cette demière s'effectue par simple adaptation de la première macro-instruction : ITER_2 utilise désormais :

— 2 variables ITE_VAL1 et ITE_VAL2

— 2 valeurs minima MIN1 et MIN2

— 2 valeurs maxima MAX1 et MAX2

— une adresse SPG de sous-programme de traitement de boucle, appelé à chaque exécution.

Une triple boucle imbriquée

L'adaptation de la boucle simple autorise la création d'une triple boucle imbriquée, suivant le même modèle que les deux précédentes : la macro ITER_3 fait appel aux paramètres suivants :

— 3 variables ITE_VAL1, ITE_VAL2 et ITE_VAL3 ;

— 3 valeurs minima MIN1, MIN2 et MIN3 ;

— 3 valeurs maxima MAX1, MAX2 et MAX3 ;

— Une adresse SPG de sous-programme de traitement de boucle.

Cette macro convient fort bien pour la réaliser d'une ou plusieurs triples boucles imbriquées ; cependant, si notre application souhaite utiliser un nombre important de ce type de boucles, une quantité mémoire considérable se trouvera monopolisée par ces dernières. Nous nous sommes en effet préoccupés jusqu'à présent de l'optimisation en vitesse des boucles, sans nous intéresser à celui de l'optimisation en mémoire.

Pour résoudre ce problème, nous utiliserons une méthode simple, qui consiste à transformer une partie de la macro-instruction de boucle en sous-programme ; nous perdrons quelques cycles en rapidité, mais gagnerons un nombre impressionnant d'octets, au niveau de l'occupation mémoire.

La macro ITE_3 synthétise l'optimisation en vitesse et en mémoire, sans perdre en lisibilité.

Un exemple d'application

Le listing fourni comporte un exemple d'utilisation d'une de nos boucles : il s'agit de la macro ITER_1, qui réaliser ici une boucle simple et affiche en majuscules les lettres de l'alphabet à l'écran. La boucle effectue une itération de ''A'' à ''Z'', par pas de 1, en exécutant à chacune des itérations le traitement TRAITO ; la variable est ITE_VAL1, dont la valeur est interceptée à chaque fois par TRAITO, qui affiche ensuite le caractère correspondant.

LES DISQUETTES DE LA REVUE
GESDOC

Logiciel de recherche documentaire (voir page .....).
Livré sur disquette 3.5'' programme et sources utilisable directement ou modifiable par vous-même.
Permet la recherche par auteur, sujet, titre, éditeur, mots-clés,...
Remplace des logiciels qui sont vendus dix fois plus cher...
Une application générée à l'aide de ATELIER !

COUPON
à adresser à La Revue de l'Utilisateur de l'IBM PC
2, rue d'Amsterdam
75009 Paris
Tél. : 47.64.07.57. - 42.67.93.80.

Nom ..... **Prénom** .....
Adresse .....
.....
Code Postal ..... **Ville** .....
Téléphone .....
Veuillez m'envoyer ..... logiciel(s) GESDOC au prix unitaire de 450 F (TTC).
Ci-joint un chèque de ..... F. En règlement à Softin Communications.

Le programme source ARTOPTI.ASM en Assembleur MASM 4.0 permet d'obtenir une fois assemblé un module objet ARTOPTI.OBJ, sur lequel l'édition de liens effectuée avec LINK.EXE produit un programme directement exécutable ARTOPTI.EXE. Lancez le programme en tapant ARTOPTI, et appuyez sur une touche à chaque fois qu'une lettre est affichée, afin de suivre pas à pas le déroulement de la boucle.

```
ARTOPTI, et appuyez sur une touche à chaque fois qu'une lettre
à-pas le déroulement de la boucle.


; ARTOPTI.ASM
;
; Efface une zone à l'écran.
;
EFFC_ZON    MACRO       LIG_H,COL_H,LIG_B,COL_B,ATT
MOV   AL,0
MOV   BH,ATT
MOV   CH,LIG_H
MOV   CL,COL_H
MOV   DH,LIG_B
MOV   DL,COL_B
MOV   AH,06H
INT   10H
ENDM
;
; Attend une touche au clavier.
;
ATTN_CLV    MACRO
MOV   AH,0
INT   16H
ENDM
;
;
; Envoie un caractère à l'écran.
;
TELE_CAR    MACRO       CAR
MOV   AL,CAR
MOV   BL,0
MOV   BH,0
MOV   AH,0EH
INT   10H
ENDM
;
;
; Initialise un MOT avec un autre MOT.
;
INIT    MACRO       MEM,VAL
MOV   AX,VAL
MOV   MEM,AX
ENDM
;
;
; Compare deux MOTS.
;
CMPA_22     MACRO       VAR1,VAR2
IFDIF   <VAR1>,<AX>
MOV   AX,VAR1
ENDIF
CMP   AX,VAR2
ENDM
;
; Effectue un branchement si deux MOTS sont égaux.
;
SI_EGAMACRO       VAR1,VAR2,ADR
'€  CMPA_22     VAR1,VAR2
JE    ADR
ENDM
;
;
SI_DIF  MACRO       VAR1,VAR2,ADR
CMPA_22     VAR1,VAR2
JNE   ADR
ENDM
;
;
SI_SUP MACRO        VAR1,VAR2,ADR
CMPA_22     VAR1,VAR2
JA    ADR
ENDM
;
;
SI_INF  MACRO       VAR1,VAR2,ADR
CMPA_22     VAR1,VAR2
JB    ADR
ENDM
;
;
SI_SUPEG    MACRO       VAR1,VAR2,ADR
CMPA_22     VAR1,VAR2
JAE   ADR
ENDM
;
;
SI_INFEG    MACRO       VAR1,VAR2,ADR
CMPA_22     VAR1,VAR2
JBE   ADR
ENDM
;
; Additionne deux MOTS.
;
ADDI    MACRO       VAR1,VAR2,SOM
MOV   AX,VAR1
ADD   AX,VAR2
MOV   SOM,AX
ENDM
;
; Boucle simple: variable en ITE_VAL1.
;
ITER_1 MACRO        MIN1,MAX1,PAS1,SPG
LOCAL V0,V1
PUSH  ITE_VAL1
INIT    ITE_VAL1,MIN1
V1:MOV      BX,OFFSET SPG
CALL  BX
ADDI    ITE_VAL1,PAS1,ITE_VAL1
SI_INFEG    ITE_VAL1,MAX1,V1
POP   ITE_VAL1
ENDM
;
;
; Boucle imbriquée: variables en ITE_VAL1 et ITE_VAL2.
;
```

```
ITER_2 MACRO        ITE_MIN1,ITE_MAX1,ITE_PAS1,ITE_MIN2,ITE_MAX2,ITE_PAS2,SPG
LOCAL V0,V1,V2
PUSH  ITE_VAL1
PUSH  ITE_VAL2
INIT    ITE_VAL1,ITE_MIN1
INIT    ITE_VAL2,ITE_MIN2
V1:CMPA_22  ITE_VAL2,ITE_MAX2
JA      V2
MOV   BX,OFFSET SPG
CALL  BX
ADDI  ITE_VAL2,ITE_PAS2,ITE_VAL2
JMP   V1
V2:CMPA_22  ITE_VAL1,ITE_MAX1
JE      V0
INIT    ITE_VAL2,ITE_MIN2
ADDI  ITE_VAL1,ITE_PAS1,ITE_VAL1
JMP   V1
V0:
POP   ITE_VAL2
POP   ITE_VAL1
ENDM
;
; Triple boucle imbriquée: variables en ITE_VAL1, ITE_VAL2 et ITE_VAL3
;
ITER_3 MACRO
        ITE_MIN1,ITE_MAX1,ITE_PAS1,ITE_MIN2,ITE_MAX2,ITE_PAS2,ITE_MIN3,ITE_MAX3,IT
E_PAS3,SPG1
LOCAL V0,V1,V2,V3
PUSH  ITE_VAL1
PUSH  ITE_VAL2
PUSH  ITE_VAL3
INIT    ITE_VAL1,ITE_MIN1
INIT    ITE_VAL2,ITE_MIN2
INIT    ITE_VAL3,ITE_MIN3
V1:SI_SUP     ITE_VAL3,ITE_MAX3,V2
MOV   BX,OFFSET SPG1
CALL  BX
ADDI  ITE_VAL3,ITE_PAS3,ITE_VAL3
JMP   V1
V2:SI_SUPEG  ITE_VAL2,ITE_MAX2,V3
INIT    ITE_VAL3,ITE_MIN3
ADDI  ITE_VAL2,ITE_PAS2,ITE_VAL2
JMP   V1
V3:SI_SUPEG  ITE_VAL1,ITE_MAX1,V0
INIT    ITE_VAL3,ITE_MIN3
INIT    ITE_VAL2,ITE_MIN2
ADDI  ITE_VAL1,ITE_PAS1,ITE_VAL1
JMP   V1
V0:
POP   ITE_VAL3
POP   ITE_VAL2
POP   ITE_VAL1
ENDM
;
; Retour au DOS.
;
RETN_DOS    MACRO
MOV   AL,0
MOV   AH,4CH
INT     21H
ENDM
;
;
CODE  SEGMENT     PUBLIC
ASSUME CS:CODE, DS:CODE, ES:CODE, SS:PILE
;
;
ITE_VAL1      DW    0
ITE_VAL2      DW    0
ITE_VAL3      DW    0
ITE_MIN1      DW    0
ITE_MIN2      DW    0
ITE_MIN3      DW    0
ITE_MAX1      DW    0
ITE_MAX2      DW    0
ITE_MAX3      DW    0
ITE_PAS1      DW    0
ITE_PAS2      DW    0
ITE_PAS3      DW    0
;
;
DEBUT:
;
MOV   AX,CODE
MOV   DS,AX
MOV   ES,AX
;
; Efface l'écran.
EFFC_ZON      0,0,24,79,07H
; Réalise la boucle.
ITER_1 "A","Z",1,TRAIT0
; Retourne au DOS.
RETN_DOS
;
; Le traitement de boucle.
TRAIT0:
;
; Récupère la valeur de la variable de boucle.
MOV   AX,ITE_VAL1
; Affiche le caractère.
TELE_CAR      AL
; Attend l' appui d'une touche.
ATTN_CLV
; Renvoie le contrôle à la boucle.
RET
;
CODE  ENDS
;
PILE    SEGMENT     STACK
DW    512 DUP(?)
PILE ENDS
;
END   DEBUT
;
```

Conclusion

L'ensemble de ces boucles en Assembleur demeure adaptable, et peut être aisément modifié en fonction du but poursuivi ; en tout état de cause, ceci constitue une illustration de la puissance des macros-instructions de l'Assembleur, qui permettent en quelque sorte de créer son propre langage - un langage dans le langage -. Les avantages fournis sont non négligeables : rapidité, compacité, et lisibilité, mais aussi complexité plus grande autorisée.

Ce dernier terrain est peut-être le vrai domaine réservé et ultra exclusif de l'Assembleur, demeurant un véritable sanctuaire de la programmation : avec les techniques et la méthodologie décrites ci-dessus, vous pourrez ainsi créer des boucles jusqu'à 24, 32 ou 48 niveaux si votre application est vraiment ambitieuse, sans encourir le risque de recevoir les fatidiques messages d'erreur "OUT OF MEMORY" ou "STRUCTURE TOO COMPLEX".

Du problème à la solution programmée...

Le poète Nicolas BOILEAU a écrit : "Avant donc d'écrire apprenez à penser". Cette citation devrait servir de règle d'or aux programmeurs, car combien sont-ils à réfléchir pleinement à leur application avant de commencer à s'abîmer les yeux devant leur écran...

Jean-Christophe RIAT

Lorsqu'il allume son ordinateur, tout bon informaticien devrait se demander s'il a fait une analyse détaillée du problème qu'il doit résoudre. Cela permet de faire une programmation modulaire et structurée qui réduit de façon considérable les risques d'erreur et accélère d'autant la phase de mise au point.

Les programmes sont alors bien plus lisibles, plus facilement modifiables et leurs performances s'en trouvent améliorées. Nous allons illustrer cette "théorie" sur un exemple précis : la réalisation en langage Turbo Pascal d'un programme de conversion de nombres sous forme numériques en français littéral.

Avant de vouloir faire résoudre un problème à la machine, il faut d'abord être capable de le solutionner soi-même, ou tout au moins de connaître un algorithme pour en venir à bout. Nous rappelons pour notre exemple, dans l'encadré ci-joint, les règles de grammaire concernant l'orthographe des adjectifs numéraux cardinaux ainsi que le vocabulaire utilisé pour "transcrire" les nombres supérieurs au million.

En analysant cette méthode de transcription, nous nous apercevons que pour lire un nombre, il est tout d'abord nécessaire de le décomposer en paquets de six chiffres, en partant des unités. Selon sa position dans le nombre, chaque paquet se voit affublé du qualificatif "million", "billion", "trillion", etc., ou sans qualificatif s'il s'agit du paquet le plus à droite, (cf. l'exemple donné dans l'encadré).

La lecture du nombre de chaque paquet se fait toujours de la même façon. On sépare les six chiffres en deux blocs de trois chiffres. Le bloc de gauche (s'il existe) indique le nombre de milliers, et celui de droite les unités. Ceci explique pourquoi, depuis toujours, l'homme a repris l'habitude de laisser un blanc ou de mettre un séparateur tous les trois chiffres dans les grands nombres...

Dans ces trois chiffres, celui de gauche correspond aux centaines, celui du milieu aux dizaines et celui de droite aux unités. Pour les centaines, on lit le chiffre s'il est différent de "1" et on lui accole le qualificatif cent. Il faut ajouter un "s" s'il y a plusieurs centaines et rien après.

Pour les dizaines et les unités, c'est plus complexe ! Une façon d'aborder le problème est de considérer que les dizaines 0, 6 et 8 ont une numérotation à 20 positions, alors que pour les dizaines 2, 3, 4, 5, elle n'est qu'à 10 chiffres. Le système numérique à 20 position comprend les états de "0" à "19" et celle à 10 positions, ceux de "0" à "9". Ainsi, l'écriture du nombre s'obtient en traduisant le chiffre des dizaines (on assimile 1 à 0, 7 à 6 et 9 à 8) suivi de la traduction de celui des unités.

Comme la langue française est très riche, il reste à définir les séparateurs entre ces deux traductions. Un tiret ou la coordination "et" peuvent être intercalés. Cette dernière s'impose lorsque les unités valent "1" et que les dizaines appartiennent à l'intervalle [2...7], (vingt et un, trente et un,...). Dans les autres cas, on mettra un tiret. Il reste encore à faire attention à l'accord du mot vingt qui prend la marque du pluriel pour le nombre "80".

Définir les limites d'utilisation...

Avant de se lancer dans l'écriture du programme, il reste encore à définir ses limites de fonctionnement. Quels sont les plus grands nombres à manipuler ? Comment l'utilisateur introduit son nombre ? Comment sortir du programme ? Etc...

Le choix a été fait de limiter le nombre à 50 chiffres, d'autoriser l'utilisateur à inclure des blancs dans le nombre introduit et de sortir du programme lorsqu'une chaîne vide est entrée au clavier.

Afin de séparer les problèmes, nous avons tout d'abord défini une procédure lecture_nombre qui retourne dans le paramètre variable "nombre", une chaîne de caractères à la syntaxe correcte. Cette procédure lit ce que l'utilisateur entre au clavier, puis vérifie que chaque caractère est bien un chiffre, en supprimant au passage les blancs éventuels.
Elle vérifie également que le nombre ne commence pas par un ou plusieurs "0" et, le cas échéant, les élimine. Le fait de supprimer les blancs et les "0" du début permet de mettre le nombre sous une forme normalisée qui pourra alors être très facilement exploitée dans la suite du programme. L'utilisateur est sollicité tant que le nombre introduit n'est pas syntaxiquement correct.

La solution informatique...

Après l'analyse, la première chose à faire est de définir la structure des données. Dans notre exemple, il est nécessaire de mémoriser dans les tables, le vocabulaire à employer pour la transcription des nombres.

Grâce à l'analyse effectuée au début, la structure de ces tables coule pratiquement de source. Quatre tableaux sont nécessaires : un pour mémoriser les multiples du million appelé littéral_million, un pour stocker les qualificatifs des dizaines nommé littéral_dizaine et deux pour archiver les unités, littéral_0_a_9 pour celles de 0 à 9 et littéral_10_a_19 pour celles de 10 à 19.

Pour rendre le programme plus lisible, le chargement de ces tables est fait dans une procédure spéciale, appelée initialisation. Le fait de partitionner l'application en différents modules permet de décomposer un problème complexe en sous-problèmes plus simples et donc plus facile à résoudre.

Ceci est encore plus vrai pour la procédure de conversion. L'analyse a montré que la conversion de tout nombre se ramenait toujours à celles d'un ou plusieurs nombre à trois chiffres. Nous définissons donc une procédure conversion_1_a_999 qui admet comme argument une chaîne de 3 décimales et affiche en retour ce nombre en français littéral.

Après avoir séparé les chiffres des centaines, dizaines et unités, la procédure traite chaque décimale séparément. Pour les centaines, l'adjectif numéral associé est affiché, s'il est différent de 1 (on dit cent et pas un cent !), grâce au tableau litteral_0_9. Puis le qualificatif "cent" ou "cents" est accolé en respectant les règles d'accord.

Pour les dizaines, on utilise le tableau litteral_dizaine, en prenant soin d'ajouter le "s" s'il s'agit du nombre "80". On complète ensuite par un séparateur qui peut être un tiret un "et" ou un blanc, selon les règles déduites de l'analyse. Les unités sont affichées grâce à l'astuce expliquée plus haut : si les dizaines valent 1, 7 ou 9 ou utilise le tableau litteral_10 _19 sinon on emploie la table litteral_ 0_a_9.

Cette procédure conversion_1_a _999 est utilisée par la procédure conversion_1_a_999999. Cette dernière admet argument une chaîne de 6 décimales et affiche en retour le nombre en français. Pour ce faire, elle sépare la chaîne argument en deux blocs de 3 chiffres et pour chacun d'eux fait appel à la procédure conversion_1_a_999.

Le qualificatif "mille" est juxtaposé au texte associé au bloc de gauche. Il faut simplement faire attention à éliminer les effets de bord : si un des blocs est nul, il ne faut pas le convertir ; si le bloc des milliers vaut "001", il ne doit pas être transcrit : on dit "mille" et pas "un mille !".

La procédure de conversion globale s'appuie sur la procédure conversion_ 1_999999. Elle commence par ajouter, si nécessaire, des "0" en tête de la chaîne à convertir afin que le nombre total de chiffres soit multiple de 6. Puis elle décompose le nombre en blocs de 6 chiffres et pour chacun d'eux fait appel à la procédure conversion_1_a _999999. Le qualificatif de multiplicité en millions, éventuellement mis au pluriel, est ajouté à côté de la traduction de chaque bloc. Comme dans le cas précédent, il faut faire attention aux effets de bord.

A l'aide de ces différentes procédures, le programme principal devient trivial à écrire. Il commence par appeler la procédure d'initialisation, puis répète l'enchaînement lecture_nombre/conversion jusqu'à ce que le nombre à convertir soit égal à la chaîne vide. Pour soigner la présentation, quelques WRITELN ont été ajoutés...

Ce qu'il faut retenir...

Au travers d'un exemple concret, nous avons mis en évidence les phases importantes que tout programmeur, digne de ce nom, se doit de respecter impérativement. L'analyse du problème, qui est bien souvent bâclée, constitue la partie la plus importante de tout projet. Une mauvaise analyse entraînera immanquablement à plus ou moins long terme des erreurs : oublis de certains cas particuliers de fonctionnement, mauvaise décomposition du problème, résolution mal ou non optimisée, etc.

Lorsque la programmation est déjà commencée, voire terminée, apporter des rectificatifs importants devient alors un vrai casse-tête ! Le temps gagné durant l'analyse est alors perdu au centuple lors des corrections du programme...

Durant la phase d'analyse, la première chose importante est de bien comprendre ce qu'il faut faire. Combien de fois voit-on une application tourner sur trois pattes à cause d'une mauvaise communication entre l'utilisateur et le programmeur ?

Une fois que l'on sait de quoi on parle, il est primordial de définir les limites de fonctionnement, c'est-à-dire les cas de figures qui peuvent se produire et ceux qui ne doivent pas arriver. Ceci fait, la recherche d'une méthode de résolution du problème peut commencer, la partie analyse doit également prendre en compte les problèmes posés par la communication entre l'application et l'utilisateur.

Lorsque l'analyse a débouché sur un algorithme complet, il faut choisir une structure de données, c'est-à-dire une méthode de stockage des informations au sein du programme. Une bonne structure de données doit être d'une exploitation aisée, facile à comprendre, et non redondante. L'organisation des données contribue pour une part importante dans la facilité d'écriture d'un programme ainsi que dans ses performances.

Avant de commencer à programmer, il est encore nécessaire de décomposer l'application en modules. Chaque module doit être défini clairement avec ses données en entrée, la tâche qu'il exécute et les variables qui sont modifiées après son exécution. Il est primordial de donner des noms explicites aux fonctions, procédures et variables du programme : "solution_equation:= resoudre(equation1)" est tout de même plus claire que "s:=r(e1) ! Toujours dans un soucis de clarté, ne pas oublier les commentaires !

Une fois tout ceci réalisé, la partie programmation doit se passer sans problème majeur. Se méfier tout de même des effets de bord (division par zéro, chaîne trop longue, variable hors intervalle, etc.) qui surviendront immanquablement ; quelques tests bien placés en viendront cependant facilement à bout. Il est recommandé de vérifier tout particulièrement ce que l'utilisateur entre au clavier.

Après la correction des inévitables erreurs de frappe et d'étourderie, le programme doit normalement fonctionner correctement. Mais les performances ne sont pas toujours celles attendues. Il peut alors être nécessaire de tout remettre en cause et de reprendre l'étude depuis le début. Mais si les différentes étapes ont été scrupuleusement suivies, les versions suivantes du projet seront réalisées nettement plus rapidement que la première !

Cet article se termine comme il s'est commencé, par une citation, cette fois-ci de Blaise PASCAL, qui est l'inventeur de la première calculatrice automatique, l'ancêtre des ordinateurs actuels : "L'homme est visiblement fait pour penser ; c'est toute sa dignité et tout son métier ; et tout son devoir est de penser comme il faut".

Un petit rappel de grammaire...

Les nombres mathématiques sont représentés en français littéral par les adjectifs numéraux. Ceux-ci se composent :

— de mots simples : un, deux, trois, quatorze, quinze, cinquante, cent, mille, etc. ;

— de mots composés : dix-sept, vingt et un, soixante-dix, quatre-vingt-dix, etc.

Il est d'usage de mettre un trait d'union pour les mots composés inférieurs à cent et non liés par la conjonction "et". On écrira par exemple : vingt et un, vingt-deux, vingt-trois, etc.

Les adjectifs numéraux sont en général invariables. Il existe cependant des exceptions sans lesquelles la langue française perdrait une grande partie de son charme ! Vingt et cent prennent la marque du pluriel quand, multipliés par un autre adjectif, ils forment le second terme d'un objet adjectif numéral composé. Mais on ne ne met plus d's s'ils sont suivis d'un autre adjectif numéral. Ainsi on écrira par exemple : deux cents, deux cent un, quatre-vingts, quatre-vingt-dix, etc.

Ces règles constituent sans doute pour beaucoup un rappel. Mais qui serait capable de lire le nombre suivant : 120 665 240 370 210 866 ? Pourtant c'est bien possible ! Il s'agit du nombre cent vint mille six cent soixante-cinq billions deux cent quarante mille trois cent soixante-dix millions deux cent dix mille huit cent soixante-dix.

En effet, selon la règle (N) légale en France depuis le 03/06/61, également utilisée en Allemagne et au Royaume-Uni notamment, on a les équivalences suivantes :

1 000 000 = 10^6 : un million.
10^{12} : un million de millions appelé un billion.
10^{18} : un million de billions appelé un trillion.
10^{24} : un million de trillions appelé un quatrillion.
10^{30} : un million de quatrillions appelé un quintillion.
10^{36} : un million de quintillions appelé un sextillion.
10^{42} : un million de sextillion appelé un septillion.
10^{42} : un million de septillions appelé un octillion.

Ces différents multiples du million prennent la marque de pluriel. Le milliard, pourtant couramment employé, qui équivaut à 1 000 millions n'a pas d'existence légale.

Pour relativiser l'utilisation de ces grands nombres, il est tout de même bon de savoir qu'un billion de Francs en billets de 500 Francs, empilés les uns sur les autres formerait une pile haute de 200 kilomètres !

```
{*************************************************************************}
{*                                                                       *}
{*  Conversion Numérique Littérale                                       *}
{*                                                                       *}
{*  Langage Turbo Pascal Versions 3, 4 ou 5                              *}
{*                                                                       *}
{*  (c) Jean-Christophe RIAT                                             *}
{*                                                                       *}
{*  Janvier 1988                                                         *}
{*                                                                       *}
{*  Ce programme permet de convertir en français courant tout nombre écrit *}
{*  sous forme numérique. Chaque nombre peut comporter jusqu'à 50 chiffres *}
{*                                                                       *}
{*************************************************************************}

PROGRAM Conversion_Numerique_Litterale;

USES Crt;  { Ligne à supprimer pour utiliser avec Turbo Pascal Version 3.xx }

TYPE

  { Différents types utilisés pour la représentation des nombres }

  nombre_1_a_999    = STRING[3];
  nombre_1_a_999999 = STRING[6];
  nombre_quelconque = STRING[50];

VAR

  { Vocabulaire littéral employé lors de la conversion }

  litteral_0_a_9   : ARRAY[0..9] OF STRING[6];
  litteral_10_a_19 : ARRAY[0..9] OF STRING[8];
  litteral_dizaine : ARRAY[2..9] OF STRING[12];
  litteral_million : ARRAY[2..9] OF STRING[15];

  { Le nombre à convertir sous sa forme numérique }

  nombre_a_convertir : nombre_quelconque;


{ ------------------------------------------------------------------------ }
{  Procédure de chargement des différents tableaux de vocabulaire littéral  }
{ ------------------------------------------------------------------------ }

PROCEDURE initialisation;

  BEGIN

    { Chargement des littéraux de 0 à 9 }

    litteral_0_a_9[0]:='zéro';          litteral_0_a_9[1]:='un';
    litteral_0_a_9[2]:='deux';          litteral_0_a_9[3]:='trois';
    litteral_0_a_9[4]:='quatre';        litteral_0_a_9[5]:='cinq';
    litteral_0_a_9[6]:='six';           litteral_0_a_9[7]:='sept';
    litteral_0_a_9[8]:='huit';          litteral_0_a_9[9]:='neuf';
```

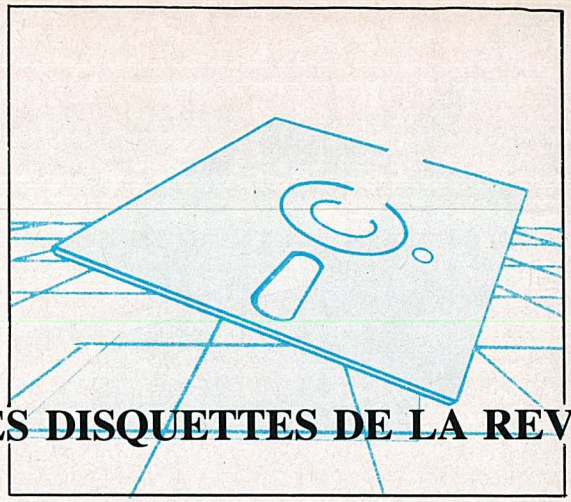


```
    { Chargement des littéraux de 10 à 19 }

    litteral_10_a_19[0]:='dix';         litteral_10_a_19[1]:='onze';
    litteral_10_a_19[2]:='douze';       litteral_10_a_19[3]:='treize';
    litteral_10_a_19[4]:='quatorze';    litteral_10_a_19[5]:='quinze';
    litteral_10_a_19[6]:='seize';       litteral_10_a_19[7]:='dix-sept';
    litteral_10_a_19[8]:='dix-huit';    litteral_10_a_19[9]:='dix-neuf';

    { Chargement des dizaines littérales }

    litteral_dizaine[2]:='vingt';       litteral_dizaine[3]:='trente';
    litteral_dizaine[4]:='quarante';    litteral_dizaine[5]:='cinquante';
    litteral_dizaine[6]:='soixante';    litteral_dizaine[7]:='soixante';
    litteral_dizaine[8]:='quatre-vingt'; litteral_dizaine[9]:='quatre-vingt';

    { Chargement des millions littéraux }

    litteral_million[2]:='million';     litteral_million[3]:='billion';
    litteral_million[4]:='trillion';    litteral_million[5]:='quatrillion';
    litteral_million[6]:='quintillion'; litteral_million[7]:='sextillion';
    litteral_million[8]:='septillion';  litteral_million[9]:='octillion';

  END;

{ ------------------------------------------------------------------------ }
{  Procédure de saisie au clavier d'un nombre entré sous forme numérique.  }
{  Après vérification de syntaxe, il est soit retourné dans la variable    }
{  'nombre' s'il est correct, soit entré à nouveau dans le cas contraire.  }
{ ------------------------------------------------------------------------ }

PROCEDURE lecture_nombre ( VAR nombre : nombre_quelconque );

  VAR
    lecture_ok : BOOLEAN;
    i : BYTE;

  BEGIN
    REPEAT

      { Lecture du nombre au clavier }

      WRITE('Entrez le nombre à convertir : ');
      READLN(nombre);

      { Vérification du nombre introduit }

      lecture_ok:=TRUE;  { Le nombre est à priori supposé correct }
      i:=1;
      WHILE (i<=LENGTH(nombre)) AND lecture_ok DO
        IF nombre[i] IN ['0'..'9']
          THEN  { On passe au caractère suivant }
            i:=i+1
          ELSE
            IF nombre[i]=' '
              THEN  { Suppression du blanc }
                nombre:=COPY(nombre,1,i-1)+COPY(nombre,i+1,LENGTH(nombre)-i)
              ELSE  { Le nombre est incorrect }
                lecture_ok:=FALSE;
```

```
    IF NOT lecture_ok
      THEN
        WRITELN('Erreur... Recommencez...')
      ELSE
        IF nombre<>''
          THEN  { Suppression des '0' éventuels au début du nombre }
            WHILE (nombre<>'0') AND (nombre[1]='0') DO
              nombre:=COPY(nombre,2,LENGTH(nombre));

  UNTIL lecture_ok;
END;


{ ------------------------------------------------------------------------ }
{  Procédure de conversion en français littéral de tout nombre de 1 à 999   }
{ ------------------------------------------------------------------------ }

PROCEDURE conversion_1_a_999 ( nombre : nombre_1_a_999 );

  VAR
    centaine,dizaine,unite : BYTE;

  BEGIN

    { Transformation des centaines, dizaines et unités en chiffres }

    centaine:=ORD(nombre[1])-ORD('0');
    dizaine :=ORD(nombre[2])-ORD('0');
    unite   :=ORD(nombre[3])-ORD('0');

    { Conversion des centaines en littéraux }

    IF centaine<>0
      THEN
       BEGIN
          IF centaine<>1
            THEN
             WRITE(' ',litteral_0_a_9[centaine]);
          IF (centaine=1) OR (dizaine<>0) OR (unite<>0)
            THEN
             WRITE(' cent')
            ELSE
             WRITE(' cents');
       END;

    { conversion des dizaines en littéraux }

    IF dizaine>=2
      THEN
       BEGIN
         WRITE(' ',litteral_dizaine[dizaine]);
          IF (dizaine=8) AND (unite=0)
            THEN
             WRITE('s');
       END;
```



```
{ ajout des séparateurs }

    IF (unite<>0) OR (dizaine IN [1,7,9])
      THEN
        BEGIN
          IF dizaine IN [0,1]
            THEN
             WRITE(' ')
            ELSE
             IF (unite=1) AND (dizaine IN [2..7])
                THEN
                  WRITE(' et ')
                ELSE
                  WRITE('-');

    { conversion des unités en littéraux }

        IF dizaine IN [1,7,9]
          THEN
           WRITE(litteral_10_a_19[unite])
          ELSE
           WRITE(litteral_0_a_9[unite]);
      END;

  END;


{ ------------------------------------------------------------------------ }
{  Procédure de conversion en français littéral de tout nombre de 1 à      }
{  999999. Cette procédure s'appuie sur la procédure conversion_1_a_999.    }
{ ------------------------------------------------------------------------ }

PROCEDURE conversion_1_a_999999 ( nombre : nombre_1_a_999999 );

  VAR
    nbre_1_a_999 : nombre_1_a_999;

  BEGIN

    { Conversion des milliers : les 3 chiffres de gauche }

    nbre_1_a_999:=COPY(nombre,1,3);
    IF nbre_1_a_999<>'000'
      THEN
        BEGIN
          IF nbre_1_a_999<>'001'
            THEN
              conversion_1_a_999(nbre_1_a_999);
          WRITE(' mille');
        END;

    { Conversion des unités : les 3 chiffres de droite }

    nbre_1_a_999:=COPY(nombre,4,6);
    IF nbre_1_a_999<>'000'
      THEN
        conversion_1_a_999(nbre_1_a_999);

  END;
```



LES DISQUETTES DE LA REVUE

DISQUETTES THÉMATIQUES		
N°	Revues	
1	10,11	Simulateur du fonct. d'un micro
2	21 à 25	Debug système et assembleur
3	17	Séquentiel indexé en BASIC
4	19 à 23	Éditeur et DAO en BASIC
5	24 à 27	Étude TURBO PASCAL (jeux, math.)
6	28	Conversion fichier DOS/WORDSTAR
7	30, 32	Étude C et PROLOG
8	30	Jeu SMILEY
9	43, 47, 49	PRSC, XPRINT et param. imprim.
10	29, 30, 33, 34	Imp. & graph., stylescript.
11	26-28, 30-37, 38	Adresses utiles + bib. souris
12	52, 53	Clavier intelligent et PASSWORD
13	40, 49	Scrolling texte + MENUDER
14	50, 51	Fenêtres Pascal + masques saisie
15	32, 34, 40	Hâchurage, Scrolling HGR, XFEN
16	48, 54	Courbes fractales
17	39, 41, 42, 43	Courbes et volumes en 3D
18	34, 35	MINITEL 1
19	36, 37	MINITEL 2
20	40, 42	MINITEL 3
21	50, 52, 53	RT/CMOS, NEWAUX, INT 44H
22	46, 48, 49	MODIF/DEPL, SAUF, BATCH inter.
23	38, 39, 41	Résidents : ZIP5, TABLES, CONTEXT
24	42, 43, 44	Assemb. & DOS (ASM, HEURE, PRSC...)
25	52, 53	Gestion de fichier en TP4
26	40, 45, 54	DEDICORE, SCAN & DEL, SAUVE
27	44, 45, 40	Étude TP. 4.0 + Fenêtrage HGR
28	48 à 54	Dossier TURBO C
29	50, 54	SECUDOS et prog. invisible
30	51, 52, 53, 55	DAO en TURBO PASCAL
31	55, 56	DBASE
32	56, 58, 59, 60...	GENECR : Générateur d'écran en T.P.
33	58, 60	Compil. interp. et désas. PASCAL

UTILITAIRES		
N°	Revues	Titres
1	40	DEDICORE : Tri sélectif des répertoires pour copier, effacer, renommer
2	41	CONTEXT : Un résident pour connaître votre configuration hard & soft
3	43	PRSC : Un résident pour rediriger et paramétrer la recopie d'écran
4	45	SCAN & DEL : Pour sélectionner et effacer vos fichiers inutiles
5	47	XPRINT : Un programme résident pour configurer l'imprimante
6	49	PARAMIMP : Un résident pour contrôler et utiliser votre imprimante
7	54	LS : Une commande DIR style UNIX
8	57	VDISK : Un disque virtuel dynamique
9	57	COPYDIR : Pour copier vos répertoires sur disquettes
10	56	RESIZE : Optimisez la taille de l'environnement de votre PC
11	58	SOSFILE : Si vous voulez récupérer des fichiers endommagés, cet utilitaire est pour vous
12	58	PCTEST : Pour tout connaître sur la configuration de votre machine
13	56, 57, 59	EXFILE : Analyseur de fichiers PASCAL
14	58, 60	HARDCOPY : Impression graphique d'écran EGA-VGA en N. & B. et nuances de gris
15	61	EDITPAL : Éditeur de palette pour EGA/VGA
16	60	XMEM : Utilisation de la mémoire étendue sous DOS
17	64	ANTIVIRUS : Traitement des virus PingPong et Jérusalem (sources et programmes)
18	70, 71, 72	XMS : OptimXMS : Optimiser la mémoire XMS

BON DE COMMANDE :
à adresser à SOFTIN COMMUNICATIONS
2 rue d'Amsterdam
75009 Paris - Tél. : 42.81.54.27

Nom .....

Adresse .....

Code Postal ..... **Ville** .....

Téléphone .....

Veuillez m'envoyer la ou les disquette (s) numéros : THÉMATIQUES (160,00 F) : .....
UTILITAIRES (90,00 F) : .....

Veuillez trouver ci-joint un chèque de..... **× 160,00 F soit**..... **Francs.**
..... **× 90,00 F soit**..... **Francs.**
Total :..... **Francs.**

en règlement à l'ordre de Softin Communications.

Date : **Signature :**

```
{ ------------------------------------------------------------------------ }
{  Procédure de conversion en français littéral de tout nombre numérique.  }
{  Cette procédure s'appuie sur la procédure conversion_1_a_999999.        }
{ ------------------------------------------------------------------------ }

PROCEDURE conversion ( nombre : nombre_quelconque );

  VAR
    nbre_1_a_999999 : nombre_1_a_999999;
    i,no_bloc,taille_pb : BYTE;

  BEGIN
    IF nombre='0'
      THEN  { Cas limite : le nombre est nul }
        WRITELN(' zéro')
      ELSE
        BEGIN

          { Ajout éventuel de '0' en début pour que le nombre de chiffres
            soit multiple de 6 }

          taille_pb:=LENGTH(nombre) MOD 6;
          IF taille_pb <> 0
            THEN
              FOR i:=1 TO 6-taille_pb
                DO nombre:='0'+nombre;

          { Décomposition du nombre en blocs de 6 chiffres }

          no_bloc:=LENGTH(nombre) DIV 6;
          REPEAT
            nbre_1_a_999999:=COPY(nombre,LENGTH(nombre)-6*no_bloc+1,6);
            IF nbre_1_a_999999<>'000000'
              THEN
                BEGIN
                  conversion_1_a_999999(nbre_1_a_999999);
                  IF no_bloc<>1
                    THEN  { Affichage du qualificatif "---million" }
                      BEGIN
                        WRITE(' ',litteral_million[no_bloc]);
                        IF nbre_1_a_999999<>'000001'
                          THEN
                            WRITE('s');
                      END
                    ELSE  { On passe à la ligne }
                      WRITELN;
                END;
            no_bloc:=no_bloc-1;
          UNTIL no_bloc=0;
        END;
  END;
```



```
{ ------------------------------------------------------------------------ }
{   Enfin le corps du programme...                                         }
{ ------------------------------------------------------------------------ }

BEGIN
  CLRSCR;
  initialisation;
  WRITELN('Conversion de nombres numériques en littéraux :');
  WRITELN('----------------------------------------------');
  WRITELN;
  WRITELN('Tapez RETURN à la place du nombre pour terminer la session...');
  REPEAT
    WRITELN;
    lecture_nombre(nombre_a_convertir);
    IF nombre_a_convertir<>''
      THEN
        conversion(nombre_a_convertir)
      ELSE
        BEGIN
          WRITELN;
          WRITELN('C''est fini...');
          WRITELN('Bonne fin de journée...');
        END;
  UNTIL nombre_a_convertir='';
END.
```

Rectificatif
A propos du programme CALC.C

Jean-Yves DEFRAYE

Le programme CALC.C figurant dans La Revue de l'Utilisateur PC n° 74, Novembre-Décembre 1991, ne tourne pas, (deux erreurs apparaissent) :

** 355: Do statement must have while in function void calcul.
** 752: Compound statement missing } in function void calcul.

J'ai donc ajouté une accolade fermante à la ligne 350 juste avant void, fonction page 5 de mon programme (1 erreur).

** 350: Do statement must have while in function void calcul.

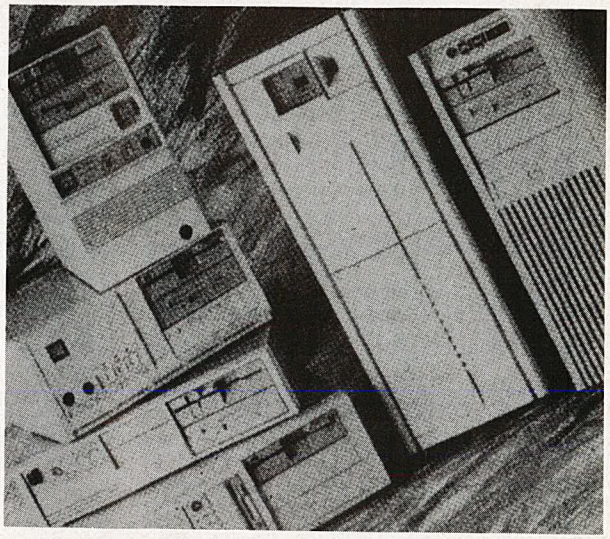
```
/*****************************************************************/
/*                                                               */
/*          PROGRAMME CALCULATRICE v4.00 (17/01/92)              */
/*                                                               */
/*                                                               */
/*                                                               */
/*                      Langage C 2.00                           */
/*                                                               */
/*****************************************************************/


#include <c:\tc\stdio.h>
#include <c:\tc\conio.h>
#include <c:\tc\math.h>

    /* variables globales */
    /* sig[1000] : tableau contenant les signes de l'expression.
       expr[1000] : tableau contenant les valeurs.
       cpo et cpf : nb de parentheses ouvertes, nb de parentheses fermees.
       dest : position dans sig[1000] de la parenthese ouverte de plus haut niveau.
       err6,err7,err4 : test des erreurs sur racines,log,division par zero.
     */

char rep=0;
int sig[1000],b=0,cpo=0,cpf=0,dest=0,pp=0,err6=0,err7=0,err4=0,inc=0;
long float expr[1000];

void presentation ()
{
clrscr();
cprintf ("\n\nBIBLIOTHEQUE DES FONCTIONS :");
cprintf ("\n\nsyntaxe                              descriptif");
cprintf ("\n\n+,-,*,/ : les operandes classiques, les mises en facteurs sont automatiques");
cprintf ("\n^ : exposant , 2^3 <enter> resultat = 8; 27^(1/3) : racine cubique de 27 = 3");
cprintf ("\nexp : exponentiel, expi = exp(1) = (expi) =2.74[....]");
cprintf ("\nlog : logarithme neperien, in10 = in(10) etc...");
cprintf ("\nsqr : racine carree, sqr4 = sqr(4) = sqr(8/2)");
cprintf ("\nsin : sinus, sini = sin(1)...");
cprintf ("\ncos : cosinus");
cprintf ("\ntan : tangente");
cprintf ("\nsinh : sinus hyperbolique");
cprintf ("\ncosh : cosinus hyperbolique");
cprintf ("\ntanh : tangente hyperbolique");
cprintf ("\nasin : arc sinus");
cprintf ("\nacos : arc cosinus");
cprintf ("\natan : arc tangente");
cprintf ("\nabs : valeur absolue, abs(2-3) = 1");
cprintf ("\nexc : arrondit par exces, exc(2.5) = 3");
cprintf ("\ndef : arrondit par defaut, def(2.5) = 2");
cprintf ("\n\nAppuyer sur une touche");
gotoxy (25,1);
cprintf ("CALCULATRICE v4.00 (17/01/92)");
rep=getch();
}

/* Affiche l'expression telle que l' ordinateur l'a comprise.
       ""      ""          au fur et a mesure des calculs. */

void afcalculs ()

{
int y=0;
    do
       {
    if (sig[y]==0 || (strchr("cstaled",sig[y])!=NULL && expr[y]!=0))
                {
                cprintf ("%.4g",expr[y]);
      sig[y]=0;
      }
```

```
        else cprintf ("%c",sig[y]);
        y++;
    }
    while (sig[y]!='E');
        y=0;
    }

        /* Fonction qui remet a jour l'expression a chaque fois qu'un calcul est
           realise. Ce reclassement supprime de l'expression la ou les valeurs qui
           ont servis a un calcul ainsi que les signes ou fonctions qui ont ete
           realisee. En outre, elle ressere l'expression autour du resultat
           calcule. Le parametre nb permet de reclasser correctement l'expression
           en fonction des elements a supprimer
           (sin,cos... = 3, asin,a cos... = 4, +, -, *, / = 2); */

    int reclasse (int nb)
    {
    inc=0;
    expr[b-nb]=expr[b];
    inc=b+1;
    do
           {
           inc++;
    expr[inc-(nb+1)]=expr[inc];
    sig[inc-(nb+1)]=sig[inc];
    }
    while (sig[inc]!='E');
    b=b-(nb+1);

       /* affiche l'expression reclassee si les resultats
       intermediaires sont demandes */

           if (rep=='o' || rep=='O')
           {
           afcalculs();
cprintf ("\n");


           }
     return (b);
    }

        /* cette fonction effectue les calculs en tenant compte des priorites
           et des parentheses */

    void calcul ()
    {
    do
    {
    b=0, dest=0;

        /* recherche du plus haut niveau de parenthese */

    do
    {
       if (sig[b]==40) dest++;
       b++;
    }
    while (dest<cpo);
    pp=b;

       /* calcul de la puissance x^y */

    do
    {
      if (sig[b]=='^')
    {
    expr[b]=pow (expr[b-1], expr[b+1]);
    b=reclasse (1);
    }

       /* calcul de la valeur absolue abs x */

if (sig[b]=='s' && sig[b-1]=='b' && sig[b-2]=='a')
           {
    expr[b]=fabs (expr[b+1]);
    b=reclasse (2);
    }

    /* test si x n'est pas negatif ou nul apres une fonction log */

    if (sig[b]=='g' && sig[b-1]=='o' && sig[b-2]=='l' && expr[b+1]<=0)
    {
    err++;
    break;
    }

    /* calcul la la fonction log x */
    if (sig[b]=='g' && sig[b-1]=='o' && sig[b-2]=='l')
           {
    expr[b]=log (expr[b+1]);
      b=reclasse (2);
      }

          /* test si x n' est pas negatif apres une fonction sqr */
    if (sig[b]=='r' && sig[b-1]=='q' && sig[b-2]=='s' && expr[b+1]<0)
    {
    err++;
    break;
    }

    /* calcul la racine carre de x : sqr x */

    if (sig[b]=='r' && sig[b-1]=='q' && sig[b-2]=='s')
    {
    expr[b]=sqrt (expr[b+1]);
      b=reclasse (2);
      }

    /* arrondit par exces */


       if (sig[b]=='c' && sig[b-1]=='x' && sig[b-2]=='e')
        {
       expr[b]=ceil (expr[b+1]);
         b=reclasse (2);
         }

       /* arrondit par defaut */

       if (sig[b]=='f' && sig[b-1]=='e' && sig[b-2]=='d')
        {
       expr[b]=floor (expr[b+1]);
         b=reclasse (2);
         }

       /* calcul de l'expotentiel exp de x */

       if (sig[b]=='p' && sig[b-1]=='x' && sig[b-2]=='e')
        {
       expr[b]=exp (expr[b+1]);
         b=reclasse (2);
         }

       /* calcul du sinus hyperbolique */

       if (sig[b]=='b' && sig[b-1]=='n' && sig[b-2]=='i' && sig[b-3]=='s')
        {
         expr[b]=sinh (expr[b+1]);
         b=reclasse (3);
         }

       /* calcul du cosinus hyperbolique */

       if (sig[b]=='h' && sig[b-1]=='s' && sig[b-2]=='o' && sig[b-3]=='c')
        {
         expr[b]=cosh (expr[b+1]);
```

```
    b=reclasse (5);
    }

/* calcul de la tagente hyperbolique */

if (sig[b]=='h' && sig[b-1]=='n' && sig[b-2]=='a' && sig[b-3]=='t')
  {
    expr[b]=tanh (expr[b+1]);
    b=reclasse (3);
    }

/* calcul du sinus et de l'arc sinus */

if (sig[b]=='n' && sig[b-1]=='i' && sig[b-2]=='s' && sig[b+1]!='h')
if (sig[b-3]=='a')
      {
expr[b]=asin (expr[b+1]);
b=reclasse (3);
       }
  else
      {
expr[b]=sin (expr[b+1]);
b=reclasse (2);
}

/* calcul du cosinus et de l'arc du cosinus */

if (sig[b]=='s' && sig[b-1]=='o' && sig[b-2]=='c' && sig[b+1]!='h')
    if (sig[b-3]=='a')
      {
expr[b]=acos (expr[b+1]);
    b=reclasse (3);
  }
 else
  {
expr[b]=cos (expr[b+1]);
    b=reclasse  (2);
}

 /* calcul de la tangente et de l' arc tangente */

if (sig[b]=='n' && sig[b-1]=='a' && sig[b-2]=='t' && sig[b+1]!='h')
if (sig[b-3]=='a')
  {
expr[b]=atan (expr[b+1]);
    b=reclasse (3);
    }
else
  {
expr[b]=tan (expr[b+1]);
    b=reclasse (2);
    }
      b++;

/* Tous ces calculs sont effectuees jusqu'a ce que l'on trouve la
     parenthese fermante ou la fin de l'expression notee 'E' */

}
 while (sig[b]!=41 && sig[b]!=69);
b=pp;
do
 {

    /* calcul de la multiplication */

    if (sig[b]==42)
     {
      expr[b]=expr[b-1]*expr[b+1];
        b=reclasse (1);
        }

/* Test si la valeur suivant une division n'est pas nulle */

    if (sig[b]==47 && expr[b+1]==0)
    {
    err4++;
break;
}

/* Calcul de la division */

    if (sig[b]==47)
    {
    expr[b]=expr[b-1]/expr[b+1];
       b=reclasse (1);
       }
    b++;
}
while (sig[b]!=41 && sig[b]!=69);
b=pp;
do
{
if (err4>0) break;

      /* Calcul de l' addition */

    if (sig[b]==43)
    {
    expr[b]=expr[b-1]+expr[b+1];
       b=reclasse (1);
       }

       /* Calcul de la soustraction */

    if (sig[b]==45)
    {
    expr[b]=expr[b-1]-expr[b+1];
       b=reclasse (1);
     b++;
}
while (sig[b]!=41 && sig[b]!=69);

       /* Suppression des parentheses inutiles */

if (cpo>0)
{
b=pp-1;
expr[b]=expr[b+1];
sig[b]=0;
inc=b+2;
do
{
inc++;
expr[inc-2]=expr[inc];
sig[inc-2]=sig[inc];
}
while (sig[inc-2]!='E');
}
if (cpo>=0)
{
cpo--;
cpf--;
}
}
while (cpo>=0);
}
}

/* Cette fonction permet d'obtenir le tableau des valeurs
   de l'equation saisie, uniquement dans le cas ou l'expression
   contient des inconnus X */

void fonction ()
{
```

Que votre volonté soit faite !

Votre micro-ordinateur sur mesure.



Solidement implantée dans le domaine de la fabrication de micro-ordinateurs depuis plus de quatre années, **CB Informatique** vous propose une configuration adaptée à vos besoins demeurant parfaitement évolutive au fil du temps.

Testés durant 72 heures dans nos laboratoires, les micro-ordinateurs CB vous assurent de la qualité professionnelle.

Le coût ? Nous vous en laissons seuls juges !

Voici donc quelques exemples de micro-ordinateurs CB standards pour lesquels nous nous ferons un plaisir de vous proposer un devis correspondant à vos choix.

Tous nos prix s'entendent H.T. départ CB.	Hercules Monochrome	V.G.A. Monochrome	V.G.A. Couleur 800 x 600
CB 286 - 12 Disque dur 40 Mo AT Bus	**5 700**	**6 200**	**7 600**
CB 386 SX - 20 Disque dur 80 Mo AT Bus	**11 200**	**11 650**	**13 900**
CB 386 - 33 Disque dur 110 Mo AT Bus	**16 100**	**16 500**	**19 900**

Toutes ces configurations comprennent : un boîtier "Desktop", 1 mégaoctet de mémoire centrale, un lecteur 5,25"(1,2 Mo) ou 3,5" (1,44 Mo), les contrôleurs d'écran et de disques, un clavier 102 touches AZERTY ainsi que deux ports série et un port parallèle.

Tarif spécial Revendeurs : nous consulter au tél : (1) 43.96.93.18 ou Fax : (1)43.96.07.44

Je souhaite recevoir une proposition concernant le matériel suivant:

Boîtier	Carte mère	Mémoire	Lecteurs	Disque dur	Écran	Souris	MS DOS 4.01	Autres
Desktop Mini-Tower Tower	*286-12, 386 SX-20 386-20, 386-25 386-33*	*1, 2, 4, 8 Mo*	*1 Lecteur 5,25 ou 3,5 2 Lecteurs 5,25 ou 3,5*	*20,40,80,110 Mo ou supérieur*	*Hercules mono VGA mono VGA coul. 800x600*	*Oui Non*	*Oui Non*	*Par exemple : streamer, imprimante, scanner...*

Société : _____
Nom et prénom : _____
Adresse : _____
Code postal : _____
Ville : _____
Téléphone / Fax : _____ / _____

CB informatique
30, avenue du Maréchal de Lattre de Tassigny
94410 SAINT MAURICE
Tél : (1) 43.96.93.18
Fax : (1) 43.96.07.44

```
int i=0,j=0,sig2[1000];
long float expr2[1000];
long float mn=0,mx=0,pre=0;
for (i=0;i<=1000;i++)
sig2[i]=expr2[i]=0;
i=0;
do
{
sig2[i]=sig[i];
expr2[i]=expr[i];
i++;
}
        while (sig[i-1]!='E');
        cprintf ("\n\n Equation y=");
        afcalculs();
        cprintf ("\n\n Domaine de definition :");
        cprintf ("\n\n Minimum : ");
        scanf ("%lf",&mn);
        cprintf ("\n Maximum : ");
        scanf ("%lf",&mx);
        cprintf ("\n Precision : ");
        scanf ("%lf",&pre);
        cprintf ("\n\n");
        j=0;
        do
{
i=cpo=cpf=0;
     if (mn>(-2e-12) && mn<(2e-12)) mn=0;
     /* remplace x dans l'equation par une valeur comprise entre
        mn et mx avec un pas de 'pre' */
        do
{
     sig[i]=sig2[i];
     expr[i]=expr2[i];
     if (sig[i]=='x' && sig[i-1]!='e' && (sig[i+1]!='p' || sig[i+1]!='c'))
{
        sig[i]=0;
       expr[i]=mn;
}
     if (sig[i]==40) cpo++;
     if (sig[i]==41) cpf++;
     i++;
}
     while (sig2[i-1]!='E');
     i=0;

     /* affichage du resultat de l'equation */

     afcalculs();
     calcul();
     if (err4!=0 || err6!=0 || err7!=0)
{
    cprintf (" = Indetermine ou Impossible\n");
     err4=err6=err7=0;
}
     else
     cprintf (" = %.14g\n",expr[0]);
     if (j>=20)
{
     cprintf ("\nAppuyez sur une touche ...\n\n");
     rep=getch();
     j=0;
}
     mn=mn+pre;
     j++;
}
     while (mn<=mx);
     if (j>1)
{
     cprintf ("\n\nAppuyez sur une touche ...");
     rep=getch();
     }
}

main ()
{
char t[1000],ky;
int a=0,x=0,y=0,f=0,err1=0,err2=0,stop=0;
    int demar=0,err3=0,err5=0,mem=0,n=0,z=0;
    do
{
    do
{
    stop=0,f=0,a=0,cpo=0,cpf=0,y=0,x=0;
    do
{
    expr[f]=sig[f]=t[f]=0;
    f++;
}
    while (f<=100);
    f=0;
    if (demar==0) presentation();
    demar=1;
    clrscr ();
    cprintf ("\nSaisissez l'expression : ");

    /* Boucle de saisie avec emulation du <DEL> qui permet mieux qu'un
       scanf de ne pas etre limitee par 255 characteres */

    do
{
    t[a]=getche();
    if (t[a]=='x')
    if (t[a-1]>47 && t[a-1]<58)
{
    t[a+1]='x';
    t[a]='*';
    a++;
}
    if (t[a]==8)
{
    if (a>0)
{
    t[a]=0;
    cprintf  (" \b");
    a--;
    if (t[a]=='x')
{
    t[a]=0;
    a--;
}
       t[a]=0;
}
    else
{
    t[a]=0;
    gotoxy (26,2);
    printf ("\a");
}
}
    if (t[a]==32) a--;
    if (t[a]!=0) a++;
}
    while (t[a-1]!=13);
    t[a-1]=0;
    a=0;

    /*Boucle de decomposition de la chaine de caracteres t[1000] contenant
    l'expression en une expression 'comprehensible' par la fonction de calcul
    cette boucle separe les signes et fonctions des valeurs, convertit les
    codes ascii des nombres en valeur recomposee.
    Elle introduit le signe '*' lors des mises en facteurs etc .. */

    do
```

MARLO le réflexe micro

ACER 1120/SX

80386SX-20 — 2 Mo RAM — 1 disque dur 40 Mo — VGA couleurs — Garantie 1 an sur site
Prix MARLO : 13 500 F

SHARP PC6661

80386SX-20 MHz — 2 Mo RAM — 1 lecteur 3''1/2 — Disque dur 60 Mo — Écran LCD VGA — DOS — Poids : 2,7 kg
Prix MARLO : 17 990 F

COMPAQ LTE 386S/20

80386SX-20 MHz — 2 Mo RAM — 1 lecteur 3''1/2 — Disque 30 Mo — Écran LCD VGA — Poids : 3,4 kg
Prix MARLO : 21 990 F
Version 60 Mo : 23 790 F

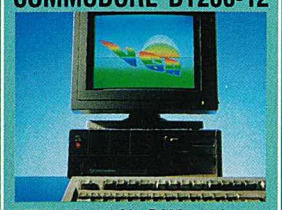
TOSHIBA T1000LE

80C86 à 9,54 MHz — 1 Mo RAM — Disque dur 20 Mo — LCD 640×400 — Poids : 3 kg — DOS
Prix MARLO : 10 500 F

TOSHIBA T2000SXe

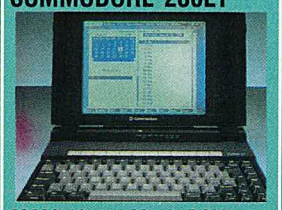
80386SX-20 — 2 Mo RAM — Disque dur 40 Mo — LCD VGA — Poids : 2,5 kg — DOS
Prix MARLO : 21 990 F
Version 60 Mo : 23 490 F

COMMODORE DT286-12



80286-12 — 1 Mo RAM — 1 lecteur 3''1/2 — Disque dur 40 Mo — Écran VGA couleurs 14'' — MSDOS — WORKS 2 + GEOWORKS
Prix MARLO : 6 300 F

COMMODORE 286LT



80286-12 — 1 Mo de RAM — 1 lecteur 3''1/2 — Disque dur 20 Mo — Écran LCD VGA — DOS — Housse-Works 2 — Autonome
Prix MARLO : 9 900 F

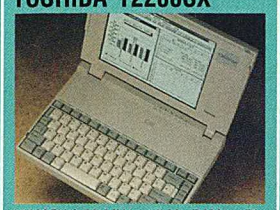
SHARP 8501

80386DX-20 — 2 Mo RAM — 1 lecteur 3''1/2 — Disque dur 100 Mo — Écran LCD VGA couleurs — DOS
Prix MARLO : 43 800 F

Hewlett Packard Deskjet 500

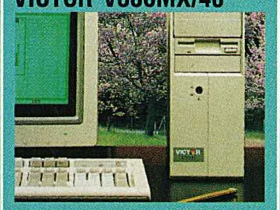
Imprimante à jet d'encre — 2 à 3 pages/minute — Garantie 3 ans dont 1 an sur site.
Prix MARLO : 3 400 F

TOSHIBA T2200SX



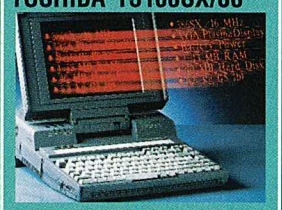
80386SX à 20 MHz — 2 Mo RAM — Disque 40 Mo — Écran LCD VGA — DOS — Poids : 2,5 kg — Autonomie : 4 heures.
Prix MARLO : 23 490 F
Version 60 Mo : 24 490 F

VICTOR V386MX/40



80386-16 — 2 Mo RAM — Disque dur 52 Mo — Écran VGA couleurs — DOS — Windows
Prix MARLO : 13 500 F

TOSHIBA T3100SX/80



80386SX-16 — 1 Mo RAM — Disque dur 80 Mo — Plasma VGA — Poids : 6,8 kg — DOS
Prix MARLO : 20 990 F

COMPAQ 386/25e

80386-25 — 4 Mo RAM — Disque dur de 60 Mo - Écran VGA couleurs
Prix MARLO : 29 700 F

TOSHIBA T3200SXC

80386SX-20 — 1 Mo RAM — Disque dur de 120 Mo — LCD VGA couleurs — Poids : 7,9 kg — DOS
Prix MARLO : 43 990 F

TOSHIBA T5200

80386-20 — 2 Mo RAM — Disque dur 100 Mo — Plasma VGA — Poids : 8,5 kg — DOS
Prix MARLO : 26 990 F
Version 200 Mo : 30 990 F

STAR LS8 III

8 ppm — 1 Mo RAM — Garantie 1 an sur site
Prix MARLO : 10 990 F
Version PostScript : 16 500 F

STAR LS4

4 ppm — 1 Mo RAM — Garantie 1 an sur site
Prix MARLO : 6 990 F
Version PostScript : 9 790 F

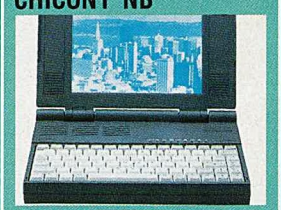
CANON LPB4+

4 pages minute — 512 Ko RAM — 300 Dpi — Cassette 50 feuilles — Garantie 1 an sur site
Prix MARLO : 5 990 F

Hewlett Packard Série III

8 ppm — 1 Mo de RAM — Garantie 1 an sur site
Prix MARLO : 11 990 F
Version PostScript : 14 790 F

CHICONY NB



80386SX-20 — 1 Mo RAM — Disque dur 40 Mo — LCD VGA — Poids : 3,4 kg — DOS — Garantie 3 ans
Prix MARLO : 12 500 F
Version 60 Mo : 14 500 F

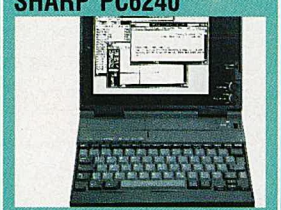
TOSHIBA T2000

80286 à 12 MHz — 1 Mo RAM — Disque 20 Mo — Écran LCD VGA — DOS — Poids : 3,1 kg
Prix MARLO : 14 990 F
Version 40 Mo : 15 890 F

Hewlett Packard Série III P

4 pages minute — 1 Mo RAM — 300 Dpi — Garantie 1 an sur site.
Prix MARLO : 7 990 F
Version PostScript : 11 195 F

SHARP PC6240



80C286 à 12 MHz — 1 Mo RAM — LCD VGA — Disque 40 Mo — Poids : 2,2 kg — DOS
Prix MARLO : 12 500 F

Tous nos prix s'entendent hors taxes T.V.A. 18,6 % en sus. Prix modifiables sans préavis.

Financement Crédit Universel

MARLO

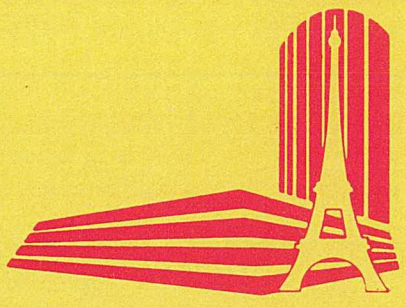
13, boulevard Ch.-de-Gaulle - 92700 COLOMBES

Tél. : (1) 47 81 20 57

Fax : (1) 47 80 59 63

OUVERT DU LUNDI AU SAMEDI

```
{

    /* Dans le cas ou l'expression contient un element du type (-2 ....
    l'algorithme introduit un 0 entre la parenthèse et  le signe car
    parenthèse -2 ne signifie rien pour la fonction calcul => erreur,
    alors que 0-2 égal bien moins 2 . */

if ((t[a-1]=='(' || a==0) && (t[a]=='+' || t[a]=='-'))
{
    sig[y+1]=sig[y];
    sig[y]=expr[y]=expr[y+1]=0;
    y++;
}

    /* test des mises en facteur.
       introduit le signe 'x' dans le cas ou l'expression contient des ')(',
       '(asin','5' */
if (t[a-1]==')' && (t[a]=='(' || t[a]=='a' || t[a]=='c' || t[a]=='s'
                    || t[a]=='t' || t[a]=='l' || t[a]=='e' || t[a]=='d'
                    || t[a]=='x' || (t[a]>47 && t[a]<58)))
{
    sig[y+1]=sig[y];
    sig[y]='x';
    expr[y]=expr[y+1]=0;
    y++;
}

    /* Algorithme de reconstitution de la partie entière d'un nombre.
       La variable mem sert à savoir si l'on se trouve juste après
       un point décimal ou pas. On reconstitue la partie entière d'un
       nombre à partir des valeur ascii des chiffres saisis, cette
       operation est effectuée pour les chiffres encadrés entre deux
       signes . X permet de savoir combien il y a de chiffres entre
       le dernier signe rencontré et le signe suivant. Ex:
       + 123). On a compté 3 chiffres a partir du '+' jusu'a la
       parenthèse. On a donc t[a-3]=49 (code ascii de 1) t[a-2]=50
       et [a-1]=51. Pour avoir 123, il faut faire (49-48)x10^2 +
       (50-48) x10^1 + (131-48)x10^0 = 1x100 + 2x10 +3x1 = 100+20+3 =
       123, CQFD. On soustrait 48, code ascii du 0 pour retrouver la
       valeur du chiffre. */

if (mem==0 && (t[a]=='(' || t[a]==')' || t[a]=='+' || t[a]=='-'
          || t[a]=='/' || t[a]=='x' || t[a]=='.' || t[a]=='^' || t[a]==0))
{
if  ((t[a-1]>=48 && t[a-1]<=57) || t[a]!=13)
{
    if (t[a]==94) t[a]=1;
    do
{
    expr[y]=expr[y]+((t[a-(x)]-48)*pow (10, (x-1)));
    x--;
}
    while (x>0);
    if (t[a]==1) t[a]=94;
    x=-1;
    n=0;
    if (expr[y]>0 &&t[a]!='.') y++;
    if (t[a]!='.')
{
    sig[y]=t[a];
    expr[y]=0;
    y++;
}
}
}

    /* Calcul de la partie decimale d' un nombre. Ce calcul est effectue
       sur le meme principe, sauf que l'on compte les chiffres a partir
       du dernier point decimal rencotre, et la reconstitution s'effectue
       a l'aide des puissances negatives de dix. */

if (mem==1 && (t[a]=='(' || t[a]==')' || t[a]=='+' || t[a]=='-'
                    || t[a]=='/' || t[a]=='x' || t[a]=='^' || t[a]==0))
{
if ((t[a]!=13) || (t[a-1]>=48 && t[a-1]<=57))
{
n--;
    x=x;
    do
{
    expr[y]=expr[y]+((t[a-(x)]-48)*pow (10, (n)));
    n--;
    x--;
}
    while (n>=(-x));
    x=-1;
    if (expr[y]>=0) y++;
    sig[y]=t[a];
    expr[y]=0;
    y++;
    mem=0;
}
}
    /* Test si un nombre ne contient pas deux points decimaux */

if (t[a]=='.' && mem==1) err5++;

    /* Si l'on rencontre un point, on sait qu'il va falloir calculer
       une partie decimale => mem=1 */

    if (t[a]=='.') mem=1;

    /* on stock les noms des fonctions dans le tableau sig[] */

    if (strchr ("scionatbhlgepxdfqr",t[a])!=NULL && t[a]!=0)
{
    x=-1;
    sig[y]=t[a];
    y++;
}

    /* Introduit le signe 'x' dans le cas des factorisations
       du type '5(' */

    if ((t[a-1]>=48 && t[a-1]<=57) && t[a-1]=='(')
{
    sig[y]=t[a];
    sig[y-1]='x';
    y++;
}
    a++;
    x++;
}
    while (t[a-1]!=0);

    /* Ajoute un 'E' pour marquer la fin de l'expression */

if (sig[y-1]!=40 && sig[y-1]!=41) sig[y-1]='E';
    else sig[y]='E';

    /* Recherche les erreurs de saisies ou mathematiques */

    b=0;
    do
{
if (sig[b]==40) cpo++;
    if (sig[b]==41) cpf++;
    if (sig[b]==40 && (sig[b+1]=='x'
                       || sig[b+1]=='/' || sig[b+1]=='^'))
       err1++;
    if (sig[b]==41 &&
        (sig[b-1]=='x' || sig[b-1]=='/' || sig[b-1]=='+'
                       || sig[b-1]=='-' || sig[b-1]=='^'))
       err2++;
```



12e Congrès de l'E.A.O. et du Multimédia

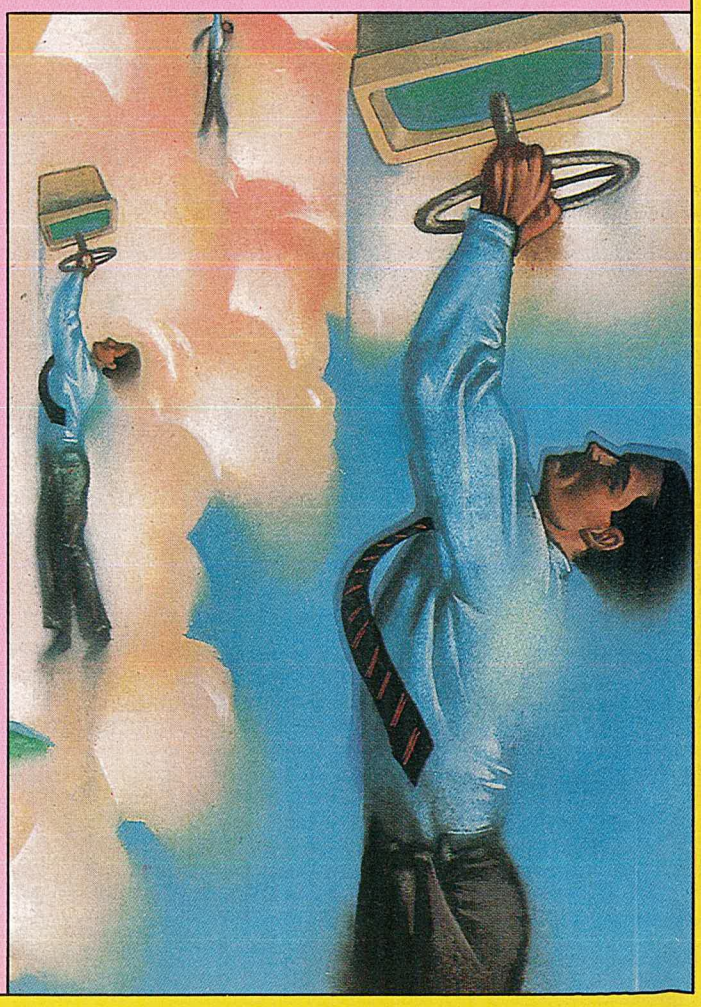
Les 12, 13 et 14 Janvier 1993
Palais des Congrès
Porte Maillot Paris 17e

Pour réserver un stand

Le Journal de la Formation et de l'E.A.O.
2 rue d'Amsterdam
75009 Paris
Tél. : 47 64 07 07 57 - 42 67 93 80

```
    if (sig[b]!=0)
   if (sig[b]!=40 && sig[b]!=41 && sig[b]!='x')
    if (strchr ("sincoatbhlgexpdfqr",sig[b+1])==NULL)
{
    if (sig[b+1]!=40 && sig[b+1]!=41 && sig[b+1]!=0)  err3++;
}
    if (sig[b]=='/' && sig[b+1]==0 && expr[b+1]==0) err4++;
    b++;
 }
    while (sig[b-1]!='E');
    if (cpo>cpf)
{
cprintf ("\nErreur 8 : il manque %d parenthese(s) fermante(s)...",(cpo-cpf));
    stop=1;
 }
    if (cpo<cpf)
{
    cprintf ("\nErreur 9 : il manque %d parenthese(s) ouvrante(s)...",(cpf-cpo));
    stop=1;
 }
   if (err1>0)
{
    cprintf ("\nErreur 1: %d Erreur(s) du type '(^ (+ (/'...",err1);
    stop=1;
 }
    if (err2>0)
{
    cprintf ("\nErreur 2: %d Erreur(s) du type '^) -) +) /) +)'...",err2);
    stop=1;
 }
    if (err3>0)
{
    cprintf ("\nErreur 3 : %d Erreur(s), deux signes sont consecutifs ...",err3);
    stop=1;
 }
    if (err4>0)
{
    cprintf ("\nErreur 4 : %d Erreur(s), division par zero ...",err4);
    stop=1;
 }
    if (err5>0)
{
    cprintf ("\nErreur 5 : %d Erreur(s), syntaxe des points decimaux...",err5);
    stop=1;
 }
    cprintf ("\n\nEXPRESSION TELLE QUE COMPRISE PAR L'ORDINATEUR :\n");
    y=0;
    afcalculs();
    if (err1!=0 || err2!=0 || err3!=0 || err4!=0 || err5!=0 || cpo<cpf || cpo>cpf)
{
    cprintf ("\n\nAppuyez sur une touche...");
    ky=getch();
 }
    b=0, x=0, y=0, err1=0, err2=0, err3=0, err4=0, err5=0, err6=0, err7=0, ky=0, dest=0;
 }
    while (stop!=0);

    /* Detecte s'il s'agit d'un calcul ou d'une equation a resoudre */

    do
{
    if (sig[x]=='x' && sig[x-1]!='e' && (sig[x+1]!='p' || sig[x+1]!='c')) ky=1;
    x++;
 }
    while (sig[x-1]!='E');
    if (x<=2) ky=2;
    if (ky==1) fonction ();
    else
{
    cprintf ("\n\nVoulez-vous voir les calculs intermediaires ? (o/n)");
    rep=getch ();
    printf ("\n\n");
    if (ky==0) calcul();
    x=0;

    /* Certaines expressions sont implcites, et l'on ne peut pas
       toujours detecter que telle operation va donner 0 ou un nombre
       negatif. C'est la raison pour laquelle des tests sont effectues pendant
       les calculs afin d'eviter un plantage. */

    if (err4>0)
{
    cprintf ("\n\nErreur 4 : %d Erreur(s), division par zero ...\n",err4);
 }
    else
    if (err6>0)
{
    cprintf ("\n\nErreur 6 : %d Erreur(s), racine negative ...\n",err6);
 }
    else
    if (err7>0)
{
    cprintf ("\n\nErreur 7 : %d Erreur(s), logarithme negatif ou nul...\n",err7);
 }
    else
{
    cprintf  ("\nLe resultat est : %.14g ",expr[0]);
 }
}
    cprintf ("\n\nVoulez-vous effectuer une autre operation ? (o\n) :");
    ky=getche ();
    err4=err6=err7=rep=0;
}
    while (ky=='o' || ky=='O');
}
```

1 fichier(s) copié(s)



N° 1 AUX ETATS-UNIS
EN VERSION FRANCAISE

Réalisez vous-même vos applications personnalisées sans taper une seule ligne de programme :

Créez des applications relationnelles de qualité et distribuez-les librement. Base de données, multi-fichiers indexés, écran multifenêtres, menus, tableaux déroulants, masques de saisie, édition de toute nature, champs calculés, conditionnels, totaux, etc...

C'est facile ! C'est étonnant !

Matériel requis : micro-ordinateur compatible PC et système d'exploitation MS-DOS, 512 K RAM, 2 MO sur disque.

Spécifications techniques : Multifichiers, 252 fichiers par application, 251 clés par fichier, 255 champs par enregistrement. Longueur enregistrement jusqu'à 64 Ko.

Taille des fichiers : illimitée.

Mode automatique : crée une application totale en quelques secondes (menus, tableaux déroulants, masques de saisie, états) à partir de la simple définition du fichier.

Aide à l'utilisation : intégrée (plus de 400 écrans).

Conversion de fichiers : import/export avec fichiers dBase, DIF, ASCII délimité.

EN CADEAU

Huit applications types, déjà développées, utilisables telles quelles ou modifiables. (Inventaire, gestion de club, mailing, facturation, répertoire téléphonique, suivi commercial, gestion scolaire).

BIONIX SOFTWARE
32, rue Molitor 75016 Paris
Tél : (33) (1) 40 71 09 36
Fax : (33) (1) 46 51 24 26

Pour la Suisse :
IHS Distribution S.A.
En Budron A 16
1052 Le Mont / Lausanne
Tél : (021) 652 34 36
Fax : (021) 652 34 52

Pour la Belgique :
Nasa Informatique S.A.
Rue Godefroid, 30 5000 Namur
Tél : (081) 22 40 77
Fax : (081) 23 19 35

LE PRIX DE LA PERFORMANCE: 990,00 F HT

PRIX DE LANCEMENT

PRIX TARIF : 2390,00 F HT.

PHOTO COMSTOCK

EDITION SPECIALE

CLARION Personnel

LE GENERATEUR D'APPLICATION UNIVERSEL

BON DE COMMANDE à remplir et à retouner à BIONIX.

Je désire recevoir CLARION Personnel au prix de lancement de 990, 00 Francs HT plus la TVA et ma participation aux frais d'envoi soit 1249,14 francs TTC.
Veuillez trouver ci-joint mon chèque de réglement à l'ordre de BIONIX.

❑ Disquettes 3 1/2 pouces
❑ Disquette 5 1/4 pouces

Signature

Nom _____ Société _____

Fonction _____ Tél _____

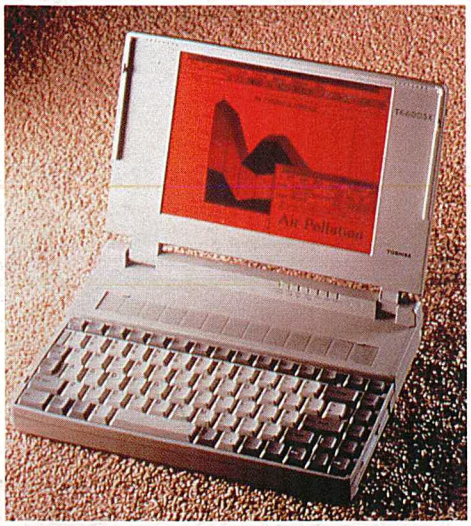
Adresse _____

_____

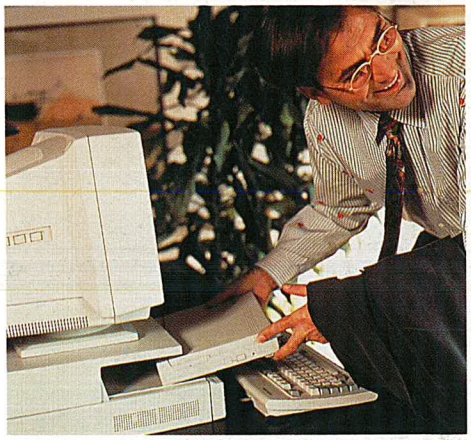
Code Postal |_|_|_|_|_| Ville _____

LA DÉDICACE INFORMATIQUE VOUS PRÉSENTE LES DERNIERS-NÉS DE TOSHIBA EN DÉTAIL

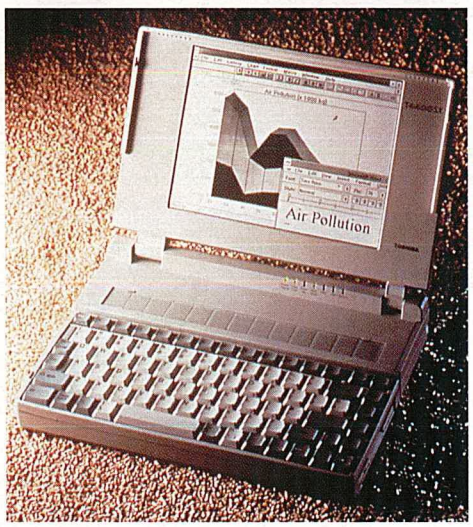
Les nouveaux notebooks Toshiba profitent des derniers progrès réalisés en matière de miniaturisation : format A4, coque en fibre de carbone, lecteur de disquettes ultra-plat 720 ko et 1,44 Mo, batteries nouvelle génération, etc. Enfin, ils sont extractibles, c'est-à-dire qu'ils s'encastrent dans la Desk Station IV aussi facilement qu'un autoradio. Celle-ci transforme les notebooks en micros de bureau complet avec écran couleur, clavier 102 touches et deux emplacements pour cartes longues.



Compouce Quadri
- Modem externe de la taille d'un briquet (poids : 90 g).
- Permet l'accès aux services vidéotex (Minitel, messageries d'entreprises, etc.), le transfert de fichiers (2400 bit/s) et l'émulation des sites centraux et des serveurs utilisant la norme VT100.
- Compatible Hayes et donc avec la plupart des logiciels du marché.



Pratique! Les nouveaux notebooks Toshiba s'encastrent facilement dans la Desk Station IV.



Deux versions du T4400 SX sont disponibles : une avec un écran VGA plasma et une avec un écran VGA LCD.

▲ Le T4400 SX
- Intel486 SX à 25 MHz, upgradable en Intel486 DX.
- 8 ko de mémoire cache.
- De 2 à 10 Mo de RAM.
- Ecran VGA plasma ou LCD.
- Disque dur de 80 Mo.
- 3,3 kg.

Le T3300 SL
- Intel386 SL à 25 MHz.
- 64 ko de mémoire cache.
- De 2 à 10 Mo de RAM.
- Ecran LCD VGA.
- Disque dur de 80 Mo.
- Emplacement 16 bits acceptant les cartes d'extension au format cartes de crédit (modem, réseau, mémoires flash, etc.) au standard PCMCIA 2.0.
- 2,7 kg.
- Disponible fin janvier 1992. ▼

Des notebooks de la taille d'une feuille A4!

▲ Le T2200 SX
- Intel386 SX à 20 MHz.
- De 2 à 10 Mo de RAM.
- Ecran LCD VGA.
- Disque dur de 40 ou 60 Mo.
- Emplacement dédié modem au format carte de crédit.
- 2,5 kg.

L'ExpressWriter 201
- La plus petite imprimante portative du monde (303 x 80 x 44 mm), compatible avec l'ensemble des logiciels du marché.
- Ultra-légère, 850 g sans la batterie.
- Autonome (une heure avec sa batterie).
- Possibilité d'imprimer sur papier thermique, sur papier standard, transparent ou même sur des enveloppes.
- Deux polices résidentes : Courier et Prestige Elite.

La Dédicace INFORMATIQUE

173, rue de Vaugirard 75015 Paris. Tél : 45 66 94 69. Fax : 43 06 35 18. Télex : 202 993